Advances in Computer Vision and Pattern Recognition

Bir Bhanu
Ajay Kumar *Editors*

Deep Learning for Biometrics

Springer

Advances in Computer Vision and Pattern Recognition

Bir Bhanu · Ajay Kumar
Editors

Deep Learning for Biometrics

Springer

Editors
Bir Bhanu
University of California
Riverside, CA
USA

Ajay Kumar
Hong Kong Polytechnic University
Hong Kong
China

ISSN 2191-6586 ISSN 2191-6594 (electronic)
Advances in Computer Vision and Pattern Recognition
ISBN 978-3-319-61656-8 ISBN 978-3-319-61657-5 (eBook)
DOI 10.1007/978-3-319-61657-5

Library of Congress Control Number: 2017943828

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

With a very security-conscious society, biometrics-based authentication and identification have become the focus of many important applications as it is widely believed that biometrics can provide accurate and reliable identification. Biometrics research and technology continue to mature rapidly, driven by pressing industrial and government needs and supported by industrial and government funding. As the number and types of biometrics architectures, sensors, and techniques increases, the need to disseminate research results increases as well.

Advanced deep learning capabilities, and deep convolutional neural networks (CNN) in particular, are significantly advancing the state of the art in computer vision and pattern recognition. The deep CNN is a biologically inspired variant of multilayer perceptron and represents a typical deep learning architecture.

Since 2006, we have organized a series of high-quality Annual Biometrics Workshops under the auspices of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR). This series has emerged as the premier forum for showcasing cutting-edge research from academia, industry, and government laboratories. During the past few years the CNN-based techniques, as evidenced by the increasing number of papers at CVPR and the biometrics workshop, have shown strong ability to learn effective feature representation from input data, especially for the perceptual and biometrics-related tasks.

The book is based on a selection of topics and authors from the proceedings of the 2016 biometrics workshop and a general call for chapters to the computer vision, pattern recognition, and the biometrics communities at large. The selection of chapters in this book is made after two rounds of rigorous review process.

Outline of the Book and Chapter Synopsis

Many of the biometrics applications require the highest level of accuracy that is being made available with the advanced deep learning capabilities. This book addresses many aspects of biometrics research issues relating to different biometrics modalities with the sole goal of improving performance for the biometrics identification. A brief and orderly introduction to the chapters is provided in the following.

The book chapters in this book on *Deep Learning for Biometrics* are organized under four major parts. **Part I** deals with deep learning for face biometrics and it includes three chapters on this topic in the book.

Chapter 1 by Grill-Spector, Kay, and Weiner on the functional neuroanatomy of face processing: insights from neuroimaging and implications for deep learning provides insightful details on the connections between currently popular deep neural network architectures for the biometrics and neural architectures in the human brain. The authors in this chapter detail a range of findings from the neuroimaging techniques on the anatomical features of the face network and the computations performed by the face-selective regions in the human brain. These empirical findings can lead to more accurate deep neural networks for face recognition.

One of the challenges related to the applications of the deep learning-based biometrics solutions is the computational complexity of the selected network. Many applications of face recognition demand highly efficient search capabilities from the large databases and in a database which requires smaller template size and faster template-matching capabilities.

Chapter 2 by Vizilter, Gorbatsevich, Vorotnikov, and Kostromov on real-time face identification via a multi-convolutional neural network and boosted hashing forest describes the development of boosted hashing-based real-time face recognition system using convolutional neural networks. This chapter presents a new biometric-specific objective function for the binary hashing that enables joint optimization of the face verification and identification.

Despite significant improvement in the accuracy of the face detection algorithms in the past decade, their accuracy is still far from that displayed by humans, particularly for the images acquired under challenging imaging environments, like with large occlusions or off-poses. Therefore, Chap. 3 by Zhu, Zheng, Luu, and Savvides on CMS-RCNN: contextual multi-scale region-based CNN for unconstrained face detection introduces a new architecture, a contextual multi-scale region-based CNN, to accurately detect human faces from a complex background and under challenging imaging conditions.

Part II of the book deals with deep learning for fingerprint, finger vein, and iris recognition. It includes three chapters that are described below.

Some of the challenging open problems in the fingerprint identification are related to accurately segmenting latent fingerprint images. Chapter 4 by Ezeobiejesi and Bhanu on latent fingerprint image segmentation using deep neural networks describes how deep neural networks can be employed to accurately segment latent fingerprint regions from complex image backgrounds. The authors present a latent fingerprint image patch and noise image patch-based classification strategy that outperforms the results on publicly available latent fingerprint databases.

Vascular biometrics identification has attracted several applications and is believed to significantly improve the integrity of biometrics systems, while preserving the privacy of an individual during authentication. Chapter 5 by Xie and Kumar on finger vein identification using convolutional neural networks and supervised discrete hashing presents a detailed deep learning-based investigation into finger vein identification. The development of competing deep learning-based solutions that can be operational using limited training data is among one of the open challenges in biometrics. Biometrics modalities like finger vein have very limited dataset in the public domain, primarily due to enhanced privacy concerns and/or due to cooperative imaging requirements. The experimental results presented in this chapter, using publicly available but limited two-session dataset indicate promises from supervised discrete hashing and provide insights into the comparative performance with state-of-the-art methods for finger vein identification.

Almost all the iris recognition systems deployed today operate using stop and stare mode. Such systems operate in constrained imaging environment and deliver remarkable accuracy. Application of iris recognition technologies for surveillance and at-a-distance applications require accurate segmentation capabilities from such images acquired with visible and near-infrared illuminations. Chapter 6 by Jalilian and Uhl on iris segmentation using fully convolutional encoder–decoder networks details the segmentation of challenging iris images using fully convolutional encoder–decoder networks.

Part III of the book deals with deep learning for soft biometrics and it includes four interesting chapters on this topic.

Accurate identification of soft biometrics features is vital for improving the accuracy of biometrics-based surveillance systems. Chapter 7 by Wu, Chen, Ishwar, and Konrad on two-stream CNNs for gesture-based verification and identification: learning user style details a deep learning framework that simultaneously leverages on the spatial and temporal information in video sequences. The experimental results presented by the authors for the identification and verification on two biometrics-oriented gesture datasets indicate results that outperform the state-of-art methods in the literature.

Developing accurate capabilities to automatically detect the soft biometrics features, like the gender, from the low resolution, off angle, and occluded face images is highly desirable for a range of biometrics applications. Chapter 8 by Juefei-Xu, Verma, and Savvides is on DeepGender2: a generative approach toward occlusion and low-resolution robust facial gender classification via progressively trained attention shift convolutional neural networks (PTAS-CNN) and deep convolutional generative adversarial networks (DCGAN). It describes the development of a deep learning-based gender classification approach. The authors describe how a progressive training strategy and the deep generative approach to recover the missing pixels can achieve excellent results for the gender classification using occluded face images.

Chapter 9 by Tapia and Aravena is on gender classification from near-infrared (NIR) iris images using deep learning. Like the previous chapter, it is also devoted to the gender classification problem but using NIR iris images. Such images are typically available during the iris acquisition and authors use a pretrained deep belief network to identify gender from these images.

Several law enforcement departments use tattoos to identify the personal beliefs and characteristics, similar to many popular soft biometrics features. Chapter 10 by Di and Patel on deep learning for tattoo recognition describes how the Siamese networks with the conventional triplet function can be used to identify tattoos in publicly available databases, with very good results.

Part IV of the book deals with deep learning for biometrics security and template protection and it consists of two chapters.

Ensuring the security of biometrics templates and the systems is an integral part of biometrics infrastructure. Chapter 11 by Pandey, Zhou, Kota, and Govindaraju on learning representations for cryptographic hash-based face template protection introduces challenges and the techniques for the protection of biometrics template. The authors in this chapter introduce template representations that are learned by CNN for the cryptographic hash-based protection of face image templates.

The last chapter in this book, i.e., Chap. 12 by Pala and Bhanu on deep triplet embedding representations for liveness detection considers widely discussed problems relating to the protection of biometrics systems against fake biometrics samples. The authors introduce a metric learning strategy that uses a variant of triplet loss function to identify fake fingerprints in image patches. Experimental results presented on publicly available database indicate outperforming state-of-the-art results from this deep learning-based strategy.

Challenges for the Future

A brief summary of the book chapters in the above paragraphs indicates that there has been significant interest in the advancement of biometrics technologies using the deep learning architectures. The work underlines challenges in deep learning when the training sample size available from a particular biometrics modality is quite small. Several authors have underlined that the factors that most influence the accuracy from deep neural networks is the depth of the network, pretraining, and the data augmentation in terms of random crops and rotations.

Several classical biometrics feature extraction and matching algorithms work very well when the images are acquired under relatively constrained environments, e.g., fingerprint and iris, with no constraints on the need for huge training samples. It is unclear how learned deep neural networks could aid, improve, or replace such popular classical methods that have been matured in past three decades, like the popular *iriscode*-based iris recognition or the minutiae matching-based fingerprint recognition widely deployed today. In this context, it is important that emerging deep learning based algorithms for biometrics should also present careful performance comparisons, on public datasets using appropriate protocols, and under open set environments or cross-dataset evaluations to make a convincing case for the benefits of biometrics community in academia, industry, and the Government.

Riverside, CA, USA
April 2017

Bir Bhanu
Ajay Kumar

Acknowledgements

The editors would like to acknowledge the support of the chapter contributors for timely submission of their high-quality research work and the reviewers for important suggestions. All the chapters went through a couple of rounds of review process which helped in the selection of final chapters. Valuable suggestions and feedback from the reviewers were provided to the authors. The incorporation of this feedback has helped in the quality of the final revised chapters in this book.

Editors would like to thank Wayne Wheeler from Springer for encouragement over the years and Simon Rees (Springer) for patience in following up the book development process. The first editor (BB) would like to acknowledge the recent support from National Science Foundation and Office of Naval Research grants related with learning and representations - DGE 0903667 video bioinformatics; IIS 0905671 video data mining; CNS 1330110 distributed sensing, learning and control; IIS 1552454 concept learning and ONR N00014-12-1-1026 continuous learning. The second editor (AK) would like to acknowledge the support of General Research Fund from the Research Grant Council of Hong Kong, via grant number PolyU 5169/13E and PolyU 152068/14E.

Riverside, CA, USA
April 2017

Bir Bhanu
Ajay Kumar

Contents

List of Figures

List of Tables

Part I
Deep Learning for Face Biometrics

Chapter 1
The Functional Neuroanatomy of Face Processing: Insights from Neuroimaging and Implications for Deep Learning

Kalanit Grill-Spector, Kendrick Kay and Kevin S. Weiner

Abstract Face perception is critical for normal social functioning, and is mediated by a cortical network of regions in the ventral visual stream. Comparative analysis between present deep neural network architectures for biometrics and neural architectures in the human brain is necessary for developing artificial systems with human abilities. Neuroimaging research has advanced our understanding regarding the functional architecture of the human ventral face network. Here, we describe recent neuroimaging findings in three domains: (1) the macro- and microscopic anatomical features of the ventral face network in the human brain, (2) the characteristics of white matter connections, and (3) the basic computations performed by population receptive fields within face-selective regions composing this network. Then, we consider how empirical findings can inform the development of accurate computational deep neural networks for face recognition as well as shed light on computational benefits of specific neural implementational features.

Introduction

Face perception is critical for normal social functioning. For example, faces provide key visual information that we use to discriminate one person from another every single day. Since face perception is ecologically and evolutionarily relevant across species [39, 52, 152, 167] a fundamental question is: *What neuroanatomical and functional features of the human brain contribute to the visual perception and recognition of faces*?

K. Grill-Spector (✉) · K.S. Weiner
Department of Psychology, Stanford University, Stanford, CA, USA
e-mail: kalanit@stanford.edu

K. Grill-Spector
Stanford Neurosciences Institute, Stanford University, Stanford, CA, USA

K. Kay
Department of Radiology, University of Minnesota, Minneapolis, MN, USA

© Springer International Publishing AG 2017
B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_1

To tackle this question, many influential theories regarding the cognitive neuroscience of face perception [12, 29, 39, 58, 67, 71, 123] have examined how regions within the brain fill the representations proposed by classical theories of face perception in cognitive psychology [14, 82, 154]. This approach has been successful in identifying a network of functional regions in the human occipito-temporal cortex that is specialized for processing faces. Indeed, many influential studies have functionally differentiated both ventral from dorsal components of this network [39], as well as functional regions from one another within these dorsal and ventral components, respectively [77, 110, 112, 125, 129, 165, 169]. This chapter focuses on the ventral component of this network and differs from prior papers and book chapters in two main ways. First, while prior articles focus on the face network as an independent system, a repetitive theme throughout this chapter is that the ventral face network is embedded within the visual system more generally. Thus, we discuss and propose that visual processing in regions outside the face network and their interaction with the face network—for example, through white matter connections—is meaningful and contributes to the efficiency of face processing. Second, in addition to understanding the functional characteristics of each region within the ventral face network, this chapter zooms into the cellular structure of neural tissue composing each region. Both of these topics are important and necessary stepping-stones toward building a mechanistic model that would inform how the anatomical structure of the face network subserves computations underlying fast and efficient face recognition.

From both neuroscience and computational perspectives, a complete mechanistic model explaining the functional neuroanatomy of face perception would (1) define each component of the ventral face network, (2) determine the anatomical features of each component, as well as their connections, (3) understand the functional characteristics (e.g., the representations and information) contained within each component, (4) derive the computations within and across components to the point that they can be modeled and cross-validated, and (5) provide an understanding regarding how anatomical features of the underlying neural circuits and their connections implement computations relevant for face perception and recognition.

This chapter synthesizes recent findings and shows that the field has made significant progress toward generating this model. First, we describe functional characteristics of the human ventral face network from well-established research findings. Second, we summarize novel macro- and microanatomical features of the ventral face network as well as features of white matter connections. Third, we discuss basic computations of the ventral face network performed by population receptive fields (pRFs). In the fourth section, we consider how recent empirical findings regarding the human ventral face network could be implemented and tested in computational models including deep convolutional neural networks (CNNs) that contain architectural features inspired by the hierarchical architecture of the ventral stream [41, 121, 136]. While deep CNN architectures are broadly “neurally inspired” by neurobiological architectural features of the ventral visual stream, they also differ from the neural architecture in the human brain in several fundamental ways (Table 1.1). Therefore, in this fourth section, we highlight some of architectural and functional features that have not yet been implemented in deep neural network architectures and consider potential computational benefits of specific functional and anatomical features of the neurobiological implementation.

1.1 The Functional Characteristics and Organization of the Ventral Face Network in the Human Brain

1.1.1 Functional Characteristics of the Ventral Face Network

Face-selective regions, which exhibit higher responses to faces compared to other stimuli have been identified with neuroimaging methods first with Positron Emission Tomography [134, 135], then with intracranial electroencephalography [1–3, 95, 116], and later with functional magnetic resonance imaging (fMRI) [72, 94, 108, 114, 150, 165]. Based on this functional characteristic, scientists identify a constellation of face-selective regions using fMRI [58]. In the occipital and temporal lobes, scientists identify face-selective regions in both ventral occipito-temporal cortex (Fig. 1.1a), as well as superior temporal cortex [39, 167]. The former are associated with face perception and recognition [35, 50, 97, 150] and the latter are associated with dynamic aspects of face perception [4, 17, 18, 110, 115, 175]. As the focus of this chapter is understanding the neural basis of face recognition, we focus on three regions of the ventral face network: IOG-faces, pFus-faces, and mFus-faces[1] (Fig. 1.1a). The former region is synonymous with the occipital face area (OFA, [42, 109]). The latter two regions are anatomically and functionally distinct components of the fusiform face area (FFA, [72, 165, 166]): pFus-faces is synonymous with FFA-1 [108] while mFus-faces is synonymous with FFA-2 [108]. Additional face-selective regions have been identified in the anterior temporal lobe [10, 70, 118, 153], but these regions are not considered part of the core face network (but see [21]) as they are not just driven by visual stimulation and are more elusive due to lower signals and susceptibility artifacts in fMRI.

The basic functional characteristic of functional regions within the ventral face network is higher neural responses to the visual presentation of faces compared to a variety of other stimuli including animate stimuli (such as limbs, bodies, and animals), familiar and unfamiliar objects, scenes, characters, and textures (Fig. 1.1b). Within each region, functional responses to face exemplars are higher than exemplars of other categories [26, 66, 98, 113] (Fig. 1.1c).

This characteristic response is maintained across sessions [13, 106, 165], tasks [15, 165], and stimulus formats (Fig. 1.1d), including photographs [65, 72], line drawings [65, 72], two-tone stimuli [24, 151], texture [36], and spatial frequency [160].

While the preferential response to faces over other stimuli is maintained across image transformations and face information can be read out from distributed

[1] See Appendix for abbreviations and definitions.

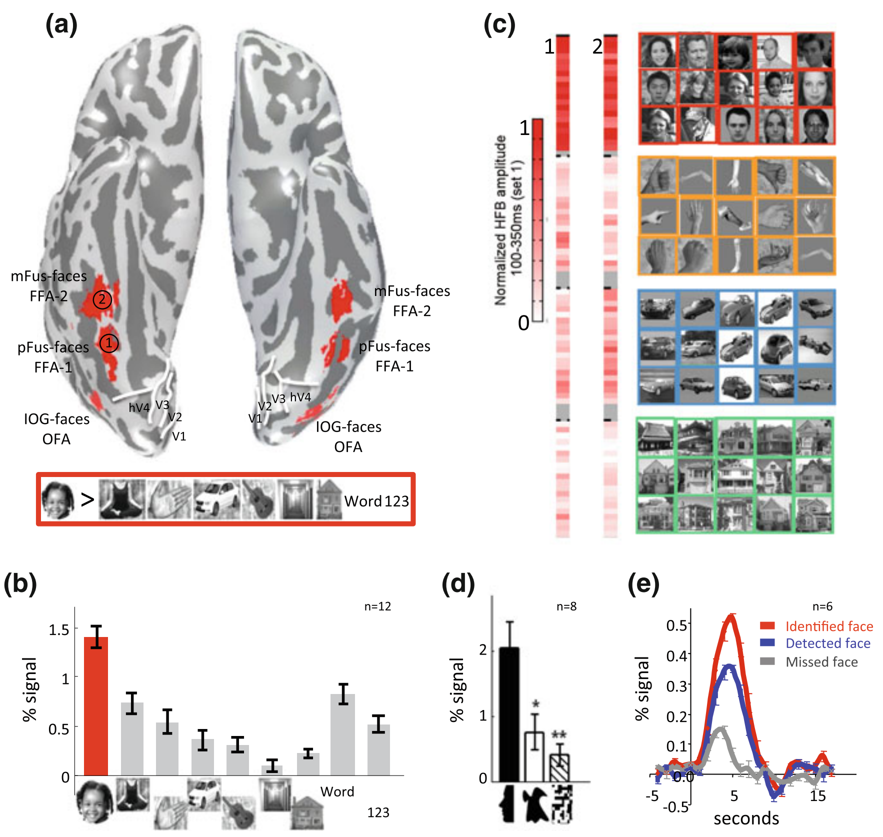


Fig. 1.1 Face-selective regions in human ventral occipito-temporal cortex. **a** Face-selective regions are identified based on higher responses to faces compared to a variety of other stimuli (faces > bodies, objects, places, and characters, t > 3, voxel level). The figure shows an inflated cortical surface of an individual subject depicting the three typical clusters of face-selective regions in ventral occipito-temporal cortex. One cluster is on the inferior occipital gyrus referred to as IOG-faces (also as occipital face area, OFA); a second cluster is on the posterior aspect of the fusiform gyrus, extending to the occipito-temporal sulcus, referred to as pFus-faces (also fusiform face area one, FFA-1); a third cluster is located about 1–1.5 cm more anterior on the lateral aspect of the fusiform gyrus overlapping the anterior tip of the mid-fusiform sulcus (MFS) and is referred to as mFus-faces (also FFA-2). *White lines* boundaries of retinotopic areas. **b** Independent analysis of response amplitudes of mFus-faces showing the typical higher responses to faces compared to other stimuli. Adapted from Stigliani et al. 2015. **c** Responses to single images in pFus- and mFus-faces. Each cell shows the normalized electrocorticography responses to single images averaged over 5 presentations and a 100–350 ms time window. The *first column* shows responses in an intracranially implanted electrode over pFus-faces/FFA-1 and the *second* shows responses from an electrode over mFus-faces/FFA-2. Responses to face images are consistently higher than responses to any of the nonface images. Adapted from [66]. **d** Responses in ventral face-selective regions to face silhouettes are significantly higher than two-tone shapes and scrambled images. Adapted from [24]. **e** Responses in ventral face-selective regions are highest when faces are identified, intermediate when they are detected but not identified, and lowest when they are missed. Adapted from [50]

responses patterns [5, 9, 19, 79, 102, 132], functional responses of the ventral face network are modulated by stimulus position [89, 183], size [183], illumination [49], contrast [126, 183], and viewpoint [49, 79, 102, 159]. For example, responses in ventral face-selective regions are higher for upright than upside down faces [24, 73, 182] and are higher for centrally presented than peripheral faces [57, 89]. Additionally, responses in face-selective regions are modulated by top-down effects, including attention [23, 25, 104], expectation [32, 143], and familiarity [6, 34, 56, 101, 164].

Critically, fMRI-adaptation [47, 49, 51] experiments have been pivotal in showing that responses in ventral face-selective regions are sensitive to face identity. For example, repeating the same face produces reduced responses due to neural adaptation [47, 49, 51, 124] and parametrically increasing the dissimilarity among face identities systematically increases the responses in face-selective regions due to release from adaptation [24, 43, 67, 90, 100]. Additionally, and consistent with behavioral aspects of face perception, (a) neural sensitivity to face identity is higher for upright than upside down faces [44] and (b) both changes in facial features and in the metric relation among features [129, 181] cause a recovery from adaptation which is associated with the perceived change in identity [26, 126, 129, 130]. Finally, neural responses to faces in ventral face-selective regions are correlated with the perception of individual participants [35, 50, 97, 150] and also causally involved in the perception of faces [68, 69, 95, 105, 111, 112, 119, 123]. For example, neural responses within mFus- and pFus-faces are low when faces are present but not detected, intermediate when faces are detected but not identified, and highest when they are identified ([50]; see Fig. 1.1e).

Altogether, these foundational studies reveal that the amplitudes of neural responses in the ventral face network are both higher for faces than nonfaces across formats and correlate with the perception of faces.

1.2 The Neural Architecture and Connections of the Ventral Face Network

1.2.1 The Functional Organization of the Face Network Is Consistent Across Participants

A striking characteristic feature of the functional architecture of the ventral face network is that the cortical location of functional regions is highly consistent across people. At the centimeter scale, face-selective regions are identifiable on specific gyri: occipital face-selective regions are located on the inferior occipital gyrus (IOG, Fig. 1.1a), while face-selective regions in ventral temporal cortex (VTC) are located on the lateral aspect of the fusiform gyrus (FG).

It is important to emphasize that gyri are not small—they are typically several centimeters long and wide and thus, have a rather large surface area. Thus, limiting a functional region to *anywhere* on these macroanatomical structures results in extensive across-subject variability with low predictability for identifying these regions from cortical folding alone [40]. However, in the last 5 years, we have gained new insights regarding the structural-functional coupling of face-selective regions at the millimeter scale. These advancements have been possible due to rigorous measurements of the variability of the cortical folding patterns of the FG and neighboring sulci [168, 170] as well as precise measurements of the relationship between functional regions and macroanatomical landmarks [99, 165, 170].

One such macroanatomical landmark to note is the mid-fusiform sulcus (MFS), which is a shallow longitudinal sulcus that bisects the fusiform gyrus (FG) into a lateral and medial portion (Fig. 1.2a). While there is anatomical variability across individuals in the length and fractionation of the MFS [170], it serves as a consistent landmark identifying functional representations. For example, irrespective of inter-individual variability in MFS morphology, the anterior tip of the MFS predicts the location of the mid-fusiform face-selective region, identifying about 80% of voxels within mFus-faces/FFA-2 (Fig. 1.2a-left [165, 170]). By comparison, the posterior tip of the MFS predicts about 50% of pFus-faces/FFA-1 (Fig. 1.2a-left). This lower functional-macroanatomical coupling is due to higher anatomical variability of the posterior compared to anterior end of the MFS. Interestingly, the structural-functional coupling extends to large-scale maps spanning several centimeters in VTC. For example, the MFS also identifies a transition within a large-scale animacy map spanning VTC [48] in which voxels that prefer animate stimuli are located lateral to the MFS and voxels that prefer inanimate stimuli are located medial to the MFS (Fig. 1.2a-center). Consequently, this consistent functional-macroanatomical coupling generates a consistent spatial relationship between multiple representations in VTC. That is, face-selective regions are consistently embedded within a larger scale representation of animate stimuli [22, 48, 93].

Given that it is more neurobiologically costly for the brain to spend the energy to generate orderly compared to disorderly representations (and not all visual representations align with anatomical axes on the cortical sheet as described above), these findings raise the following questions: (1) Are there anatomical constraints that contribute to the consistency of this functional architecture? (2) Is there a computational benefit to the regular spatial topography of functional representations in VTC?

1.2.2 The Cytoarchitecture of Face-Selective Regions

What might explain the predictable spatial arrangement of both fine-scale clusters and large-scale functional representations relative to macroanatomical landmarks? Recent evidence indicates that anatomical constraints may underlie the predictable topologies in the VTC.

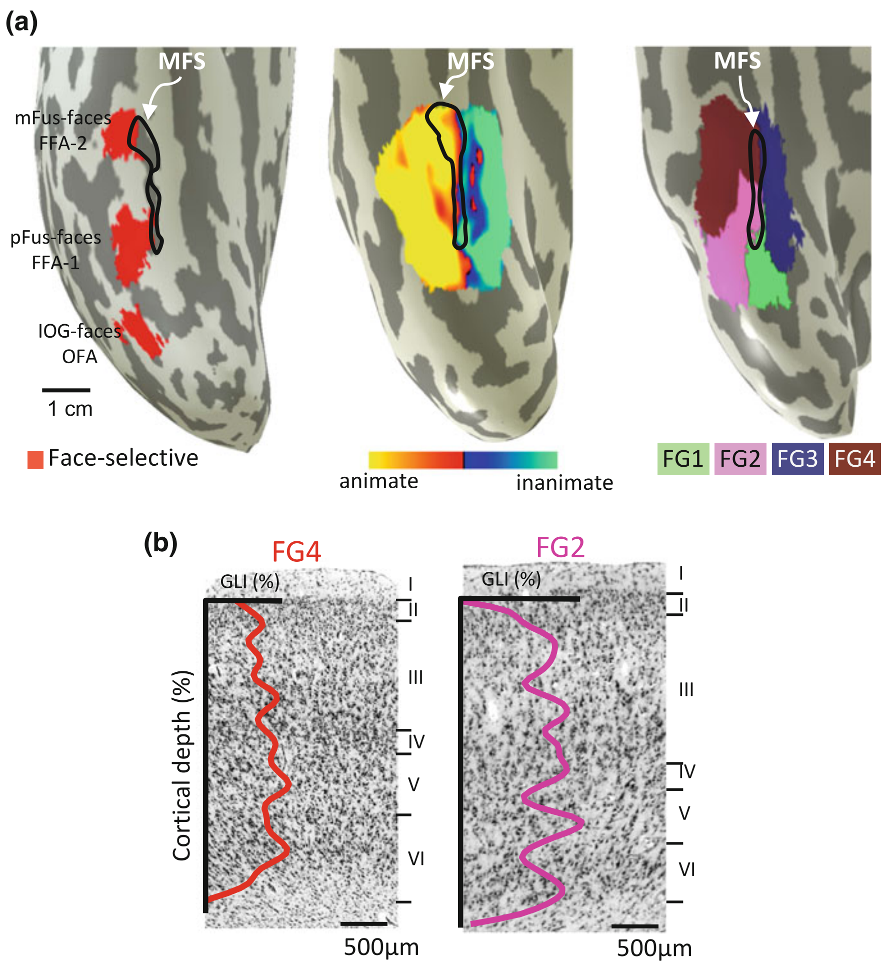


Fig. 1.2 Regular spatial structure of functional and anatomical parcellations of ventral temporal cortex. **a** Multiple representations are aligned to the mid-fusiform sulcus (MFS). Data are shown on an inflated cortical surface zoomed on ventral temporal cortex (VTC) of the right hemisphere of a representative brain. MFS is indicated by the *white outline*. *Left* Face-selective regions (faces > other categories; t > 3, voxel level, *red*) mFus-face/FFA-2 and pFus-faces/FFA-1 are predicted by the anterior and posterior tips of the MFS, respectively. *Center* MFS serves as a boundary between distributed representations of animate and inanimate representations. *Right* The MFS serves as a boundary separating lateral cytoarchitectonic regions FG4 and FG2 from medial cytoarchitectonic regions FG3 and FG1, respectively. Cytoarchitectonic areas are indicated with separate *colors* (see legend). *FG*: fusiform gyrus. **b** Histological slice showing the cell body staining and the gray level index (GLI, *line*) across cortical layers from a representative 20-micron slice through FG4 (*left*) and FG2 (*right*). There are different cell distributions and cell sizes across cortical layers between the two example slices

1.2.2.1 The Cytoarchitecture of the FG

The size, shape, and organization of cells across the six-layered cortical ribbon (the combination of which is referred to as cytoarchitecture) is a well-established criterion to parcellate the brain into areas because differences in cellular structure across cortical layers are believed to be indicative of specialized neural hardware that is utilized for particular brain functions.

In the last 4 years, four cytoarchitectonic areas have been discovered in VTC using data-driven, observer-independent techniques, which are blind to cortical folding. In the posterior aspect of the fusiform gyrus (FG) and neighboring sulci, there are two cytoarchitectonic areas referred to as FG1 and FG2 [20]. FG1 is located on the medial aspect of the FG and is characterized by a columnar structure, while FG2 is situated on the lateral aspect of the FG and has a higher cell density than FG1. Anteriorly, there are two additional cytoarchitectonic areas referred to as FG3 and FG4 ([92]; Fig. 1.2a-right). Many architectural features differentiate FG3 from FG4. For example, FG3 is characterized by a compact and dense layer II, while FG4 is characterized by a less dense layer II compared to FG3, as well as large layers III and V. Interestingly, the MFS not only serves as a landmark identifying face-selective regions and functional transitions in large-scale maps, but also serves as a landmark identifying microarchitectural boundaries separating medial fusiform cytoarchitectonic areas (FG1/FG3) from lateral fusiform cytoarchitectonic areas (FG2/FG4, Fig. 1.2a-right). Specifically, the cytoarchitectonic transition between FG1, medially, to FG2, laterally, occurs 5.41 ± 1.6 mm from the posterior MFS, while the cytoarchitectonic transition between FG3, medially, and FG4, laterally, occurs $1.42 \pm .54$ mm from the anterior MFS (Fig. 1.2a-right). Since the MFS predicts both functional and anatomical transitions in the human brain, it is natural to ask: Is there relationship between functional regions and cytoarchitectonic areas in the fusiform gyrus?

1.2.2.2 The Relationship Between FG Cytoarchitectonic Areas and the Ventral Face Network

Quantifying the relationship between cytoarchitectonic areas and face-selective regions is challenging because cytoarchitectonic areas are delineated in postmortem brains, while face-selective regions are defined in living brains. Presently, it is impossible to relate these cortical divisions within the same individual. Nevertheless, it is possible to quantitatively relate these structures by aligning them to a common cortical reference frame. This is done using cortex-based alignment, which leverages cortical folding patterns to align one brain to another, irrespective if the brains are from living or postmortem individuals [38, 122]. Implementing this novel approach revealed that functionally defined face-selective regions within the FG are cytoarchitectonically dissociable. That is, different face-selective regions are located within different cytoarchitectonic areas: mFus-faces/FFA2 is largely within FG4 ($81\% \pm 24\%$, mean $\pm$ standard deviation), while pFus-faces/FFA1 is largely within FG2 ($49.5\% \pm 24\%$) and not in other cytoarchitectonic areas of the FG [173]. These results suggest that microanatomical properties contribute to the macroanatomical positioning of mFus-faces/FFA-2 and pFus-faces/FFA-1 (even though they are both located on the fusiform gyrus). For example, pFus-faces/FFA-1 displays features of FG2 (Fig. 1.2b), which has a conspicuous layer III with larger pyramidal cells than those of mFus-faces/FFA-2, as well as a prominent and dense layer IV compared to mFus-faces/FFA-2, which has a thin and moderately dense layer IV.

These results have three important theoretical ramifications. First, distinct cytoarchitecture is evidence for differential neural hardware optimized for specialized computations. Thus, it is likely that the cytoarchitectonic differences between mFus-faces/FFA-2 and pFus-faces/FFA-1 are reflective of different computations implemented by these regions. Second, as cytoarchitectonic differences are used to parcellate brain areas, our data suggest that distinct functional regions corresponding to pFus- and mFus-faces, respectively, have distinct cytoarchitectonic structure. Third, since IOG-faces is located outside the FG, it does not overlap with any of the FG cytoarchitectonic areas. This suggests that IOG-faces/OFA is also cytoarchitectonically distinct from pFus-faces/FFA-1 and mFus-faces/FFA-2.

Together these findings suggest that the brain may have different neural hardware for specific computations implemented in each of the three faces-selective regions of the ventral face network. Nevertheless, future research is necessary to elucidate the computations that are produced by this elaborate microcircuitry as well as detailed properties of this circuitry including the precise cell types, their connections, and their 3D structure.

1.2.3 White Matter Connections of the Ventral Face Network

In addition to local cytoarchitecture, there is consensus that white matter connections also constrain the function of the brain [147, 158, 184]. Recent evidence has begun to elucidate the nature of white matter connections of the face network with four main findings. First, ventral face-selective regions IOG-, pFus-, and mFus-faces are highly interconnected with direct white matter connections [54, 117, 171]. Second, longitudinal white matter tracts connect early visual retinotopic areas located outside the face network to ventral face-selective regions [54, 80, 171]. Third, vertical white matter tracts connect dorsal stream visual regions located outside the face network to ventral face-selective regions. For example, portions of the vertical occipital fasciculus (VOF; [146, 172, 179]) connect a retinotopic region in the posterior intraparietal sulcus (IPS-0) and pFus-faces [171]. Fourth, there are distinct white matter tracts associated with the ventral face network compared to networks associated with processing other domains. For example, long-range white matter connections of pFus- and mFus-faces are distinct from white matter connections of a place-selective region in the collateral sulcus (CoS-places/PPA; [45, 117, 128, 149]).

A schematic summarizing these white matter connections is shown in Fig. 1.3. We note that this diagram is incomplete because it does not provide information regarding (1) the entire connectivity of the ventral face network, (2) the direction of

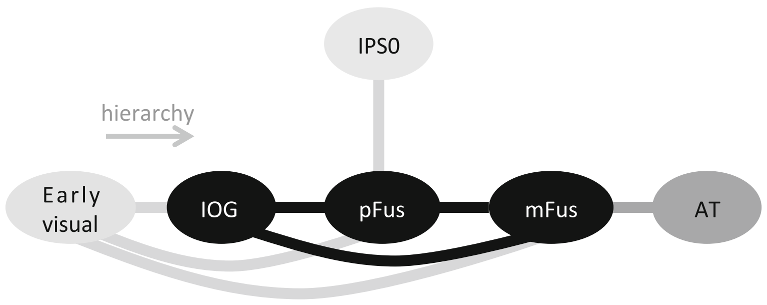


Fig. 1.3 Schematic diagram of white matter tracts of the ventral face network. *Black ovals* indicate core face-selective regions and *gray ovals* indicate regions that are considered external to the core face network. The schematic is arranged such that the hierarchical axis is from left to right. *Acronyms*: face-selective regions: *IOG*: inferior occipital gyrus; *pFus*: posterior fusiform; *mFus*: mid-fusiform: *AT*: anterior temporal; *IPS0*: a region in the intraparietal sulcus that is part of the attention network

connections (though research in animals suggests that they are bidirectional [37]), or (3) the functionality of these white matter tracts.

Nevertheless, the schematic diagram provides four important insights regarding the connectivity features of the ventral face network. First, these findings illustrate hierarchical connections from IOG- to pFus-faces and from pFus- to mFus-faces, which may provide the connectivity scaffolding for hierarchical features of the ventral face network described in the following section. Second, the network contains redundant connections: there are multiple white matter tracts reaching each region. Third, there are bypass connections that do not follow a strict hierarchical organization. For example, mFus-faces, which is considered a later stage of the processing hierarchy following pFus-faces, is connected not only to the preceding stage (pFus-faces), but also to IOG-faces and early visual cortex. Finally, there are vertical white matter tracts connecting IPS-0, which is thought to be part of the attention network [64, 137, 138, 144], to pFus-faces. These vertical tracts may facilitate top-down processing [74].

One important functional feature of redundant and bypass connections is that they may provide network resiliency in the face of injury or disease. For example, recently we had the unique opportunity to measure the face network before (1 month and 4 days) and after (1 and 8 months) a surgical resection of the IOG in a patient (SP) who underwent surgery to treat intractable epilepsy [171]. Surprisingly, downstream regions remained functionally intact despite the resection of the IOG, which would not have been predicted by a strict hierarchical organization [58]. Interestingly, this resiliency of FG face-selective regions is also reported in patients with long-term ($>$10 years) damage to the inferior occipital cortex [129, 140, 142]. By measuring white matter connections in SP, we identified the longitudinal and vertical tracts discussed above suggesting that these tracts may contribute to the resiliency of the face network after resection by enabling signals to reach these downstream regions using alternate routes from early visual cortex and/or parietal cortex [171].

While identifying white matter tracts of the ventral face network is a major stepping-stone, we recognize that future work is necessary to uncover many remaining unknowns regarding the connectivity of the ventral face network including: (1) What is the functional utility of white matter connections both within the face network as well as to regions outside the network? (2) What is the contribution of white matter connection to the spatial segregation of face-selective regions of the ventral face network? (3) What is the contribution of white matter connections to behavior?

1.3 Computations by Population Receptive Fields in the Ventral Face Network

As described in Sect. 1.2 above, the field has accrued a considerable body of knowledge regarding the functional characteristics of ventral face-selective regions, the anatomical composition and connectivity of regions within the ventral face network, and their role in perception. However, how underlying features contribute to the computations of each region and the network as a whole remains elusive. In this section, we describe progress in understanding basic computations performed across the ventral stream by population receptive fields (pRFs).

A logical starting point for developing computational models of the ventral face network is to determine *receptive field* properties (the region of the visual field within which a stimulus elicits a response from a neuron) for neurons in ventral face-selective regions for three reasons. First, receptive fields are a fundamental aspect of the processing performed by neurons in the visual system [53, 63]. Since neurons with similar RFs are spatially clustered and fMRI measures the population response of all neurons within each brain voxel (volume pixel), we can measure the *population receptive field (pRF)*—the region of the visual field that drives the population of neurons within a voxel [31, 76, 77, 162]. Second, face recognition is thought to require spatial integration across facial features rather than processing isolated facial features [148, 155, 156]. Thus, determining the location and size of pRFs in the ventral face network may inform our understanding of which parts of the face are processed in different stages of the face network. Third, understanding pRFs may shed light on fixation/viewing behavior. For example, when asked to recognize a face, participants typically fixate on the center of the face and the eyes, but when asked to judge the emotion of the face, people also fixate on the mouth [107]. These fixation behaviors suggest that the spatial capacity of face processing is not only limited, but may also be task dependent.

1.3.1 pRF Measurements Reveal a Hierarchical Organization of the Face Network

Recently, we performed a series of experiments in which we built *encoding models* (computational models that explicitly identify a set of features and computations that predict evoked brain responses) that characterize pRFs in the ventral face network. Subjects were scanned while viewing faces at different positions and sizes that systematically tiled the central visual field (12.5°). From these data, we obtained the amplitude of fMRI response of each voxel as a function of face position and size. Then, for each voxel we fit a model that predicts the response by computing the overlap of the stimulus with a 2D Gaussian, followed by a compressive nonlinearity [76, 77]. This model-based analysis (1) provides estimates of pRF properties such as size and eccentricity (distance from fixation), (2) allows comparison of pRF properties across the ventral stream hierarchy, and (3) provides insight into how space is represented in the ventral face network.

PRF mapping shows that voxels in the ventral face network are substantially modulated by the location and spatial extent of faces in the visual field, and our simple pRF model explains these modulations well. PRFs in the ventral face network illustrate several characteristics. First, pRF centers are located in the contralateral visual field [60, 77]—a finding that is consistent with retinotopic organization of visual cortex more generally [62]. That is, pRF centers of voxels in the right hemisphere are centered in the left visual field and vice versa. Second, pRFs in the ventral face network tend to be located close to the center of gaze [77, 176] rather than distributed across the visual field as in early and intermediate retinotopic areas (V1-hV4, Fig. 1.4b). Third, the average pRF size progressively increases from V1 to subsequent retinotopic areas (V2-hV4) and into the face network (IOG-, pFus- and mFus-faces; [77, 176], Fig. 1.4a). This progressive increase of average pRF size is consistent with a hierarchical organization [157]. Fourth, pRF size linearly increases with eccentricity in the face network (Fig. 1.4c; [77, 176] as in earlier regions [162]). Additionally, the slope of this relationship increases across the visual hierarchy. Consequently, the size of pRFs in the ventral face network is larger than their eccentricity. Thus, pRFs in the ventral face network always cover the center of gaze (fovea) and extend into the ipsilateral visual field, processing information from both right and left visual fields.

An intuitive illustration of the visual information processed by pRFs across the ventral processing stream is shown in Fig. 1.4d. Here, we show example pRFs at 1° eccentricity across several regions spanning the ventral stream; these pRFs are superimposed on a face sized to approximate typical conversational distance [91, 96]. The figure illustrates that a V1 pRF processes local information, such as the corner of the eye, while a pRF within hV4 may process an entire facial feature, such as an eye. However, the large and foveal pRFs in the face network process information across multiple facial features from both sides of the face. These data suggest a potential explanation for fixation behavior: when people fixate faces, they attempt to

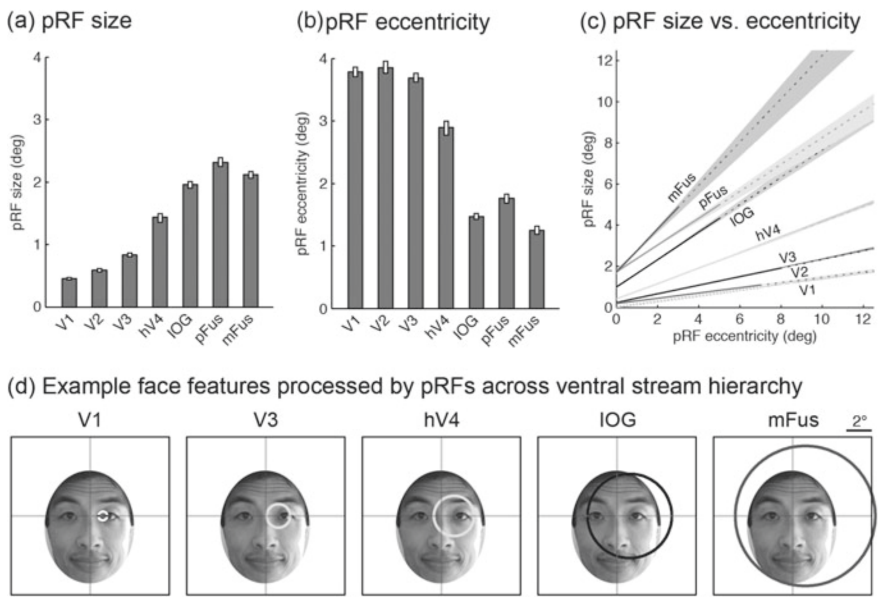


Fig. 1.4 pRFs reveal a hierarchical organization of the ventral face network. **a** Median pRF size for each area from V1 to mFus-faces (*error bars* indicate 68% confidence intervals, CIs). pRF size is defined as the standard deviation of a 2D Gaussian that characterizes the response of the pRF to point stimuli. **b** Median eccentricity of pRF centers for each area. **c** Relationship between pRF size and eccentricity (*shaded area* indicates a 68% CI on a line fitted to the data). **d** Facial features processed across the hierarchy. *Circles* indicate pRFs at 1° eccentricity (as derived from panel **c**). Each *circle* is drawn at +/− 2 pRF sizes. The depicted face is sized to simulate a conversational distance of 1 m (approximately 6.5° based on average male head sizes [91, 96]). Ascending the hierarchy, spatial information is integrated across increasingly large regions of the face, until the latest stages where entire faces are processed by neural populations within a voxel. Adapted from [77, 167]

position pRFs in face-selective regions in order to optimally integrate information across facial features.

Data comparing pRFs in the ventral face network between developmental prosopagnosics (DPs) and typical participants support the idea that spatial integration across facial features obtained by large and central pRFs is necessary for face perception [176]. DPs are individuals without brain damage, but who are impaired at face recognition without showing other visual or cognitive deficits [8, 11, 28, 30, 88]. PRF measurements reveal that pRFs in the face network (and hV4) of DPs are smaller, and rarely extend to the peripheral or ipsilateral visual field compared to typical controls. Notably, across both typicals and DPs, face recognition ability is positively correlated with pRF size in the ventral face network: participants with larger pRFs perform better than those with smaller pRFs ($r(15) = 0.63$, $p<0.007$). In contrast, face recognition ability does not correlate with pRF size in early retinotopic areas. These data provide empirical evidence suggesting that smaller pRF sizes in DPs may reflect a deficit in spatial integration, consequently affecting face recognition.

1.3.2 Attention Modulates pRF Properties, Enhancing Peripheral Representations Where Visual Acuity Is the Worst

Responses in the ventral face network are not just driven by the stimulus, but are also modulated by internal top-down goals [7, 104, 143, 180]. We characterized how top-down factors modulate pRF properties and spatial information by measuring pRFs under different attentional states [77]. PRFs were estimated separately for different tasks using identical visual stimulation, including a *digit task* during which attention

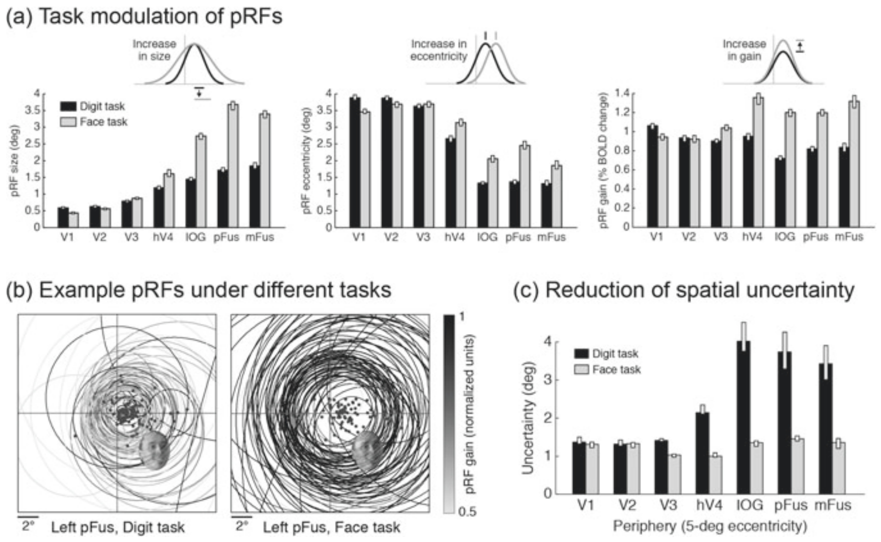


Fig. 1.5 Attention modulates pRF properties in the ventral face network, enhancing spatial representations. PRFs were measured under different tasks using the same stimulus. For the data in this panel, subjects either performed a one-back task on centrally presented digits (digit task) or a one-back task on the presented face (face task) while fixating centrally. **a** Task-induced changes in pRF properties (*bars* indicate median across voxels; *error bars* indicate 68% CIs). pRFs in IOG-, pFus-, and mFus-faces, and hV4 are larger (*left*), more eccentric (*middle*) and have increased gain (*right*) during the face task (*gray*) compared to the digit task (*black*). **b** Tiling of visual field by 100 randomly selected pRFs from *left* pFus-faces (*dots* indicate pRF centers; *circles* indicate pRFs drawn at +/− 2 pRF sizes). An example face is shown at 5-deg eccentricity. **c** Spatial uncertainty in discrimination of stimulus positions. *Bars* indicate amount of uncertainty for reference positions at 5-deg eccentricity (median across angular positions +/− 68% CI). During the digit task, uncertainty in IOG-, pFus-, and mFus-faces is large. However, during the face task, uncertainty is substantially reduced and is commensurate with spatial uncertainty in V1. Adapted from [77, 167]

was directed toward digits presented centrally, at fixation, and a *face task,* during which attention was directed toward faces, which appeared at various locations tiling the visual field (while fixating, subjects performed a one-back judgment on digits and faces, respectively).

Fitting the pRF model separately to brain responses observed under different tasks, we found that pRFs in the ventral face network and hV4 are dependent on the task. In these regions, pRFs are larger (Fig. 1.5a-left), located more peripherally (Fig. 1.5a-center), and have a higher gain (Fig. 1.5a-right) when participants attended to the faces compared to when they attended to digits presented at fixation. In contrast, pRFs in early visual areas V1–V3 were relatively stable and did not substantially change across tasks (Fig. 1.5a).

To obtain an intuitive understanding of the effect of attentional modulation of pRFs in the ventral face network, we visualize the collection of pRFs from left pFus-faces measured in the digit task (Fig. 1.5b-left) and in the face task (Fig. 1.5b-right). As pRFs are larger and more eccentric in the face than digit task, there is extended coverage of the peripheral visual field during the face task compared to the digit task. For example, while a face presented at 5° eccentricity from fixation is processed by only a handful of pRFs during the digit task, it is processed by many pRFs during the face task.

To interpret the change in spatial representations across tasks, we used a model-based *decoding approach* (inferring information from distributed patterns of brain activity across a collection of pRFs) and quantified the spatial uncertainty of the location of a face presented at 5° eccentricity from responses of a collection of pRFs spanning each visual area. Results show that the spatial uncertainly in decoding the location of the face from the collection of pRFs in the face network substantially decreases from the digit to face task (Fig. 1.5c). Surprisingly, the spatial uncertainty in the face network during the face task is similar to that obtained by V1 pRFs which are considerably smaller. These results illuminate another aspect of spatial coding: what determines the spatial resolution of processing by a collection of pRFs is not only their size, but also their scatter. In other words, large and partially overlapping pRFs may provide similar spatial precision as small and nonoverlapping pRFs, consistent with the notion of coarse coding [139, 174].

Our research of pRF properties is only a first stepping-stone of building accurate encoding models of the face network, as we have implemented a rudimentary spatial filter that performs only spatial summation, and the same type of spatial filtering throughout the ventral stream. Thus, important future elaborations of the pRF model would be to (1) include additional dimensions to the model that explain computations related to other aspects of the stimulus (e.g., its shape and/or features), and (2) determine whether and how additional pRF properties vary within each cortical region and across regions. For example, implementing an array of orientation selective filters for each V1 voxel, rather than just Gaussian filters, provides better predictive power in explaining V1 responses to natural images [75]. In the domain of face processing, elaborating pRFs is necessary for explaining basic response properties of face-selective regions not explained by the present pRF model, such as preferential responses and tuning to individual faces [46, 84, 103] and face parts [27, 61].

1.4 Eyes to the Future: Computational Insights from Anatomical and Functional Features of the Face Network

1.4.1 What Is the Computational Utility of the Organized Structure of the Cortical Face Network?

The empirical findings reviewed here reveal an organized and reproducible implementation of a neural processing system for face perception in the human brain. Since generating an organized structure is more effortful than a disorganized structure, it is possible that certain principles reflecting optimized computational strategies are produced from this functional architecture over the course of evolution or development. Therefore, computational insights can be gleaned from the specific features of the physical implementation of the ventral face network in the brain. Here, we highlight some of the architectural and functional features that have not yet been implemented in computational models and consider putative computational aspects of these features that can be tested in future research.

For example, an important aspect of computational models that can be further developed is explicitly modeling how anatomical features may contribute to the formation of dynamic, task-dependent pRFs. The present pRF approach simply treats each task as a distinct entity and estimates a model of the stimulus representation separately for each task. This provides useful insight, but further research is necessary to identify the neural mechanisms that underlie the source of the task modulations originating from other brain areas. To better understand how attention may modulate pRFs, one could consider the characteristics of white matter connections of the face network. For example, top-down connections from cortical regions outside the face network (such as the connection from IPS-0 to pFus-faces) may provide a route for top-down information to flow from parietal regions involved in attentional gating to face-selective regions. Thus, dynamic pRFs may be an outcome of an interaction between a static pRF generated by bottom-up connections and an attention field [81, 120, 141] mediated by top-down connections from IPS-0. Developing new encoding models that incorporate these anatomical features may reveal insight into the interplay between top-down and bottom-up processing in the face network—a topic that has been elusive thus far.

In addition to macroscopic anatomical features such as white matter connectivity, we believe there is also utility in incorporating microanatomical features into computational models. For example, cytoarchitectonic differences between pFus- and mFus-faces suggest that these regions have different neural hardware that may be optimized to perform different types of computations. These data therefore suggest that computational models should not necessarily implement a single generic neural computation or filter type that is duplicated across processing stages. Instead, there are likely different specialized computations occurring at different stages of processing. Furthermore, an ambitious and interesting direction for future work is to forge explicit links between anatomical properties such as cytoarchitecture, receptor architecture, myeloarchitecture, cell types, and microcircuit connections relative to the computational properties of neurons in the ventral face network.

1.4.2 What Can Deep Convolutional Networks Inform About Computational Strategies of the Brain?

Another promising avenue for future research will be to incorporate recent neuroscience findings into computational models implementing deep convolutional neural networks (CNNs; [41, 67, 85, 121, 136]), which have seen significant recent advancement. In brief, deep CNNs are neural networks composed from a series of stacked layers, in which the first layer performs operations on the input image and subsequent layers perform operations on the output of the prior layer. As such, CNNs have a feedforward architecture. Additionally, layers in a CNN typically alternate between layers performing linear operations (e.g., convolution), layers performing nonlinear operations (e.g., ReLU), and pooling layers. After several stacked layers of this sort, there are often one or more fully connected layers. The layers that perform linear operations typically contain multiple arrays of filters in which each filter performs an operation on a local region in the visual input (akin to computations by RFs in the human visual system), and the same filters are repeated across locations tiling the entire visual field. Additionally, filters in subsequent layers pool information from a local neighborhood from the prior layer, which yields an overall increase in pooling moving up the CNN hierarchy. Once the architecture of the deep CNN is built, the network is then trained to perform a task (e.g., face recognition). During training, the weights of the connections between layers are altered typically using a backpropagation algorithm [87, 127], which functions to reduce the error between the network output and the desired answer. After training, the weights are no longer changed and the processing in deep CNNs is strictly feedforward.

Deep CNN architectures are appealing because (1) they are inspired by architectural features of the ventral visual stream [41, 121, 136], (2) their performance reaches human-like performance in complex object recognition tasks [83, 177] including face recognition [145], and (3) they can predict—to a noteworthy level of accuracy—experimentally measured responses in the primate and human ventral stream [16, 33, 55, 67, 78, 86, 178]. We suggest that CNNs can be used as a tool (e.g., through simulations and analysis) to understand the computations being performed within the face network. However, as a first step, it is important to consider in what aspects the artificial architecture of deep CNNs is similar to or different than the neurobiological architecture of the human ventral visual stream.

Several aspects of deep CNNs are similar to the architecture of the human ventral visual stream (Table 1.1). For example, both systems implement: (1) computations by local filters, (2) hierarchical processing across a series of stages, (3) feedforward

Table 1.1 Computational hypotheses generated from comparisons between deep convolutional neural networks (CNNs) and neural architecture

Functional/ architectural property	Deep CNN	Human Brain	Hypothesized Computational Value
pRFs/filters			
Local computations	✔	✔	Parallel processing Distributed information
pRF/filter size increases along hierarchy	✔	✔	Useful features Invariance
pRF/filter size increases with eccentricity	✖	✔	Solution to limited brain size
Dynamic pRFs	✖	✔	Task-optimized processing
Representations			
Spatial topography	✖	✔	Efficiency and speed
Clustering of neurons with similar features	✖	✔	Reduce wiring length
Neural Hardware			
Differential neural across regions	✖?	✔?	Computation-optimized hardware
Connections			
Hierarchical connections	✔	✔	Hierarchical processing
Redundant connections	✔	✔	Resiliency to loss/damage
Bypass connections	✖	✔	Resiliency/speed
Top-down connections	during learning	✔	Learning/attentional gating

computations, (4) linear–nonlinear operations, (5) redundant connections, and (6) modification of weights during training.

While deep CNN architectures are broadly "neurally inspired" by neurobiological architectural features of the ventral visual stream, they also differ from the neural architecture in the human brain in several fundamental ways (Table 1.1). As examples, we note four differences between current deep CNNs such as AlexNet [85]) or FaceNet [131] and what is known about the ventral visual stream. First, after the training stage, filters in CNNs are fixed rather than dynamic, but in the brain, pRFs are dynamic and can be modified by top-down attention. Second, filters in CNNs are identical in size across the visual field, but in the brain, pRFs are larger in the periphery than the center of gaze. Third, in a given layer of a CNN there is no significance to the spatial arrangement of filters in a given layer, but the visual system exhibits a spatial topography of representations across spatial scales of the cortical sheet that is reproducible across individuals. Fourth, in standard CNNs, beyond the training stage, there is typically no influence of top-down connections or bypass routes, but both types of connections are present and are used by the brain. By modifying the CNN architecture in order to implement features that more accurately reflect neurobiologically plausible brain architectures, it might be possible to explore and better understand the computational benefits of specific neurobiological features (Table 1.1 – right column). Below, we give examples of three such tests.

First, the computational architecture of most current deep CNNs such as AlexNet [85] or FaceNet [131] is strictly serial, which does not respect the biological reality that the ventral face network contains bypass routes that skip stages of the processing hierarchy. Therefore, incorporating recent empirical findings of bypass routes into the architectures of CNNs may (1) make CNNs more accurate models of the human visual system and (2) could advance our understanding regarding hypothesized benefits of specific architecture features such as bypass routes. For example, the hypothesis that bypass connections provide resiliency to cortical damage could be tested by comparing the effect of a virtual lesion to an intermediate layer of a strictly serial neural network versus a virtual lesion in a non-serial neural network containing bypass connections.

Second, deep CNNs contain fixed filters (beyond the training stage) and do not incorporate top-down connections from other processing stages. Consequently, processing in deep CNNs is purely stimulus-driven and therefore, does not account for known empirical effects of task and attention on responses in the ventral face network, such as attention-modulated pRFs. Thus, one could adapt a present deep CNN like FaceNet [131] to include top-down connections in order to test whether this addition explains task and attention effects on pRF properties and if so, if it improves the efficiency and/or performance of the network.

Third, as described in Sect. 1.2.1, there is a regular spatial topography of functional representations in the human brain. This topography is evident in all stages of the visual processing hierarchy—from an orderly structure of pRF arrangement across the cortex generating spatial maps of the visual field in early and intermediate visual areas [133, 161, 163], to the object form topography [22, 59] and regularity of face-selective regions relative to macroanatomical landmarks [165, 170] in ventral temporal cortex. In contrast to the spatial regularity of fine-scale functional regions and large-scale representations in the brain, there is (1) no spatial organization to nodes in a particular layer of a CNN and (2) neurobiological costs (such as wiring length) are not explicitly accounted for by CNNs, whereas biological systems are affected by those costs. Thus, future research can examine (1) what architectonical constraints need to be added to a deep CNN for it to develop a spatial structure (perhaps guided by the cytoarchitectonic and connectivity structure of face-selective regions summarized in Sect. 1.2 and (2) what computational benefit does cortical organization provide? For example, does spatial topography enhance CNN speed or efficiency?

1.5 Conclusions

As described in this chapter, neuroimaging research has advanced our understanding regarding the functional architecture of the human ventral face network. For example, the scale of our understanding of the spatial arrangement of regions in the ventral face network has improved from centimeters to millimeters. Recent research has linked this consistent spatial arrangement of the ventral face network to underlying microanatomical features such as cytoarchitecture, as well as white matter connectivity. Mechanistically, the development of new methods deriving population receptive fields has begun to elucidate computational principles of the ventral face network. While there are key questions that remain unanswered, these new research directions have opened exciting new opportunities for understanding how the functional neuroanatomical features of the face network contribute to computations underlying human face perception. Importantly, incorporating these recent findings in up-to-date computational models will further advance the field by providing enhanced understanding of the computational benefits of specific implementational features of the human brain by integrating such features into state-of-the-art deep convolutional neural network architectures.

Acknowledgements We thank Jesse Gomez for comments and edits of an earlier draft. This research has been funded by the following grants: NSF GRFP DGE-114747, 1RO1EY02231801-A1, and 1R01EY02391501-A1. Portions of this chapter were previously included within a paper published in the Annual Reviews of Vision Science, 2017 (http://www.annualreviews.org/doi/10.1146/annurev-vision-102016-061214).

Appendix: Abbreviations and Definitions

Collateral sulcus (CoS): a primary sulcus in human ventral temporal cortex; the medial boundary of the fusiform gyrus

Cytoarchitecture: cellular organization across the six-layered cortical ribbon; a property used to parcellate brain areas from one another

Developmental prosopagnosia (DP): an impairment in recognizing faces despite normal vision, intelligence, and socio-cognitive abilities and no history of brain damage

Fusiform face area (FFA): once considered a homogenous face-selective area, it contains (at least) two cytoarchitectonically and functionally distinct components

Fusiform gyrus (FG): a hominoid-specific macroanatomical structure in ventral temporal cortex that contains (at least) four cytoarchitectonic areas and multiple functional regions

FG1–4: Labels for four cytoarchitectonic areas in the fusiform gyrus and neighboring sulci

Inferior occipital gyrus (IOG): a gyrus that is posterior-lateral to the fusiform gyrus; considered the first processing stage of the ventral face network

Mid-fusiform sulcus (MFS): a shallow, longitudinal sulcus bisecting the fusiform gyrus; a landmark identifying cytoarchitectonic and functional boundaries

mFus-faces/FFA-2: a face-selective region overlapping the anterior-lateral tip of the mid-fusiform sulcus, located within cytoarchitectonic area FG4, and 1–1.5 cm anterior to pFus-faces/FFA-1

Occipito-temporal sulcus (OTS): a primary sulcus in human ventral temporal cortex; the lateral boundary of the fusiform gyrus

Parahippocampal place area (PPA): a place-selective region in the collateral sulcus and parahippocampal cortex

pFus-faces/FFA-1: a face-selective region typically overlapping the posterior-lateral tip of the mid-fusiform sulcus, located within cytoarchitectonic area FG2, and 1–1.5 cm posterior to mFus-faces/FFA-2

Population receptive field (pRF): in fMRI, the region of visual space that stimulates a voxel

Receptive field (RF): the region of the visual field which elicits a response from a neuron

Ventral Temporal Cortex (VTC): a cortical expanse spanning the inferior aspect of the temporal lobe containing high-level visual regions involved in "what" perception and recognition.

References

1. T. Allison, H. Ginter, G. McCarthy, A.C. Nobre, A. Puce et al., Face recognition in human extrastriate cortex. J. Neurophysiol. **71**, 821–825 (1994)
2. T. Allison, G. McCarthy, A. Nobre, A. Puce, A. Belger, Human extrastriate visual cortex and the perception of faces, words, numbers, and colors. Cereb. Cortex **4**, 544–554 (1994)
3. T. Allison, A. Puce, D.D. Spencer, G. McCarthy, Electrophysiological studies of human face perception. I: potentials generated in occipitotemporal cortex by face and non-face stimuli. Cereb. Cortex **9**, 415–430 (1999)
4. T.J. Andrews, M.P. Ewbank, Distinct representations for facial identity and changeable aspects of faces in the human temporal lobe. Neuroimage **23**, 905–913 (2004)
5. S. Anzellotti, S.L. Fairhall, A. Caramazza, Decoding representations of face identity that are tolerant to rotation. Cereb. Cortex **24**, 1988–1995 (2014)
6. G. Avidan, M. Behrmann, Functional MRI reveals compromised neural integrity of the face processing network in congenital prosopagnosia. Curr. Biol. **19**, 1146–1150 (2009)
7. G. Avidan, I. Levy, T. Hendler, E. Zohary, R. Malach, Spatial vs. object specific attention in high-order visual areas. Neuroimage **19**, 308–318 (2003)
8. G. Avidan, U. Hasson, R. Malach, M. Behrmann, Detailed exploration of face-related processing in congenital prosopagnosia: 2. Functional neuroimaging findings. J. Cogn. Neurosci. **17**, 1150–1167 (2005)

9. V. Axelrod, G. Yovel, Hierarchical processing of face viewpoint in human visual cortex. J. Neurosci. **32**, 2442–2452 (2012)
10. V. Axelrod, G. Yovel, The challenge of localizing the anterior temporal face area: a possible solution. Neuroimage **2013**(81), 371–380 (2013)
11. M. Behrmann, G. Avidan, Congenital prosopagnosia: face-blind from birth. Trends Cogn. Sci. **9**, 180–187 (2005)
12. M. Behrmann, D.C. Plaut, Distributed circuits, not circumscribed centers, mediate visual recognition. Trends Cogn. Sci. **17**, 210–219 (2013)
13. M.G. Berman, J. Park, R. Gonzalez, T.A. Polk, A. Gehrke et al., Evaluating functional localizers: the case of the FFA. Neuroimage **50**, 56–71 (2010)
14. V. Bruce, A. Young, Understanding face recognition. Br. J. Psychol. **77**(Pt 3), 305–327 (1986)
15. L. Bugatus, K.S. Weiner, K. Grill-Spector, Task differentially modulates the spatial extent of category-selective regions across anatomical locations. Neuroimage (2017) (in press)
16. C.F. Cadieu, H. Hong, D.L. Yamins, N. Pinto, D. Ardila et al., Deep neural networks rival the representation of primate IT cortex for core visual object recognition. PLoS Comput. Biol. **10**, e1003963 (2014)
17. A.J. Calder, A.W. Young, Understanding the recognition of facial identity and facial expression. Nat. Rev. Neurosci. **6**, 641–651 (2005)
18. A.J. Calder, J.D. Beaver, J.S. Winston, R.J. Dolan, R. Jenkins et al., Separate coding of different gaze directions in the superior temporal sulcus and inferior parietal lobule. Curr. Biol. **17**, 20–25 (2007)
19. T. Carlson, H. Hogendoorn, H. Fonteijn, F.A. Verstraten, Spatial coding and invariance in object-selective cortex. Cortex **47**, 14–22 (2011)
20. J. Caspers, K. Zilles, S.B. Eickhoff, A. Schleicher, H. Mohlberg, K. Amunts, Cytoarchitectonical analysis and probabilistic mapping of two extrastriate areas of the human posterior fusiform gyrus. Brain Struct. Funct. **218**, 511–526 (2013)
21. J.A. Collins, I.R. Olson, Beyond the FFA: the role of the ventral anterior temporal lobes in face processing. Neuropsychologia **61**, 65–79 (2014)
22. A.C. Connolly, J.S. Guntupalli, J. Gors, M. Hanke, Y.O. Halchenko et al., The representation of biological classes in the human brain. J. Neurosci. **32**, 2608–2618 (2012)
23. T. Cukur, A.G. Huth, S. Nishimoto, J.L. Gallant, Functional subdomains within human FFA. J. Neurosci. **33**, 16748–16766 (2013)
24. N. Davidenko, D.A. Remus, K. Grill-Spector, Face-likeness and image variability drive responses in human face-selective ventral regions. Hum. Brain Mapp. **33**, 2234–2249 (2012)
25. I. Davidesco, M. Harel, M. Ramot, U. Kramer, S. Kipervasser et al., Spatial and object-based attention modulates broadband high-frequency responses across the human visual cortical hierarchy. J. Neurosci. **33**, 1228–1240 (2013)
26. I. Davidesco, E. Zion-Golumbic, S. Bickel, M. Harel, D.M. Groppe et al., Exemplar selectivity reflects perceptual similarities in the human fusiform cortex. Cereb. Cortex **24**, 1879–1893 (2014)
27. B. de Haas, D.S. Schwarzkopf, I. Alvarez, R.P. Lawson, L. Henriksson et al., Perception and processing of faces in the human brain is tuned to typical feature locations. J. Neurosci. **36**, 9289–9302 (2016)
28. B. Duchaine, K. Nakayama, The Cambridge face memory test: results for neurologically intact individuals and an investigation of its validity using inverted face stimuli and prosopagnosic participants. Neuropsychologia **44**, 576–585 (2006)
29. B. Duchaine, G. Yovel, A revised neural framework for face processing. Annu. Rev. Vis. Sci **1**, 393–416 (2015)
30. B.C. Duchaine, K. Nakayama, Developmental prosopagnosia: a window to content-specific face processing. Curr. Opin. Neurobiol. **16**, 166–173 (2006)
31. S.O. Dumoulin, B.A. Wandell, Population receptive field estimates in human visual cortex. Neuroimage **39**, 647–660 (2008)
32. T. Egner, J.M. Monti, C. Summerfield, Expectation and surprise determine neural population responses in the ventral visual stream. J. Neurosci. **30**, 16601–16608 (2010)

33. M. Eickenberg, A. Gramfort, G. Varoquaux, B. Thirion, Seeing it all: convolutional network layers map the function of the human visual system. Neuroimage (2016)
34. M.P. Ewbank, T.J. Andrews, Differential sensitivity for viewpoint between familiar and unfamiliar faces in human visual cortex. Neuroimage **40**, 1857–1870 (2008)
35. F. Fang, S. He, Cortical responses to invisible objects in the human dorsal and ventral pathways. Nat. Neurosci. **8**, 1380–1385 (2005)
36. R. Farivar, O. Blanke, A. Chaudhuri, Dorsal-ventral integration in the recognition of motion-defined unfamiliar faces. J. Neurosci. **29**, 5336–5342 (2009)
37. D.J. Felleman, D.C. Van Essen, Distributed hierarchical processing in the primate cerebral cortex. Cereb. Cortex **1**, 1–47 (1991)
38. B. Fischl, M.I. Sereno, R.B. Tootell, A.M. Dale, High-resolution intersubject averaging and a coordinate system for the cortical surface. Hum. Brain Mapp. **8**, 272–284 (1999)
39. W. Freiwald, B. Duchaine, G. Yovel, Face processing systems: from neurons to real-world social perception. Annu. Rev. Neurosci. **39**, 325–346 (2016)
40. M.A. Frost, R. Goebel, Measuring structural-functional correspondence: spatial variability of specialised brain regions after macro-anatomical alignment. Neuroimage **59**, 1369–1381 (2012)
41. K. Fukushima, Neocognitron: a hierarchical neural network capable of visual pattern recognition. Neural Netw. **1**, 119–130 (1982)
42. I. Gauthier, P. Skudlarski, J.C. Gore, A.W. Anderson, Expertise for cars and birds recruits brain areas involved in face recognition. Nat. Neurosci. **3**, 191–197 (2000)
43. S. Gilaie-Dotan, R. Malach, Sub-exemplar shape tuning in human face-related areas. Cereb. Cortex **17**, 325–338 (2007)
44. S. Gilaie-Dotan, H. Gelbard-Sagiv, R. Malach, Perceptual shape sensitivity to upright and inverted faces is reflected in neuronal adaptation. Neuroimage **50**, 383–395 (2010)
45. J. Gomez, F. Pestilli, N. Witthoft, G. Golarai, A. Liberman et al., Functionally defined white matter reveals segregated pathways in human ventral temporal cortex associated with category-specific processing. Neuron **85**, 216–227 (2015)
46. C. Gratton, K.K. Sreenivasan, M.A. Silver, M. D'Esposito, Attention selectively modifies the representation of individual faces in the human brain. J. Neurosci. **33**, 6979–6989 (2013)
47. K. Grill-Spector, R. Malach, fMR-adaptation: a tool for studying the functional properties of human cortical neurons. Acta Psychol. (Amst) **107**, 293–321 (2001)
48. K. Grill-Spector, K.S. Weiner, The functional architecture of the ventral temporal cortex and its role in categorization. Nat. Rev. Neurosci. **15**, 536–548 (2014)
49. K. Grill-Spector, T. Kushnir, S. Edelman, G. Avidan, Y. Itzchak, R. Malach, Differential processing of objects under various viewing conditions in the human lateral occipital complex. Neuron **24**, 187–203 (1999)
50. K. Grill-Spector, N. Knouf, N. Kanwisher, The fusiform face area subserves face perception, not generic within-category identification. Nat. Neurosci. **7**, 555–562 (2004)
51. K. Grill-Spector, R. Henson, A. Martin, Repetition and the brain: neural models of stimulus-specific effects. Trends Cogn. Sci. **10**, 14–23 (2006)
52. C.G. Gross, J. Sergent, Face recognition. Curr. Opin. Neurobiol. **2**, 156–161 (1992)
53. C.G. Gross, D.B. Bender, C.E. Rocha-Miranda, Visual receptive fields of neurons in inferotemporal cortex of the monkey. Science **166**, 1303–1306 (1969)
54. M. Gschwind, G. Pourtois, S. Schwartz, D. Van De Ville, P. Vuilleumier, White-matter connectivity between face-responsive regions in the human brain. Cereb. Cortex **22**, 1564–1576 (2012)
55. U. Guclu, M.A. van Gerven, Deep neural networks reveal a gradient in the complexity of neural representations across the ventral stream. J. Neurosci. **35**, 10005–10014 (2015)
56. R.J. Harris, G.E. Rice, A.W. Young, T.J. Andrews, Distinct but overlapping patterns of response to words and faces in the fusiform gyrus. Cereb. Cortex **26**, 3161–3168 (2016)
57. U. Hasson, I. Levy, M. Behrmann, T. Hendler, R. Malach, Eccentricity bias as an organizing principle for human high-order object areas. Neuron **34**, 479–490 (2002)

58. J.V. Haxby, E.A. Hoffman, M.I. Gobbini, The distributed human neural system for face perception. Trends Cogn. Sci. **4**, 223–233 (2000)
59. J.V. Haxby, M.I. Gobbini, M.L. Furey, A. Ishai, J.L. Schouten, P. Pietrini, Distributed and overlapping representations of faces and objects in ventral temporal cortex. Science **293**, 2425–2430 (2001)
60. C.C. Hemond, N.G. Kanwisher, H.P. Op de Beeck, A preference for contralateral stimuli in human object- and face-selective cortex. PLoS One **2**, e574 (2007)
61. L. Henriksson, M. Mur, N. Kriegeskorte, Faciotopy-A face-feature map with face-like topology in the human occipital face area. Cortex **72**, 156–167 (2015)
62. G. Holmes, Disturbances of vision by cerebral lesions. Br. J. Ophthalmol. **2**, 353–384 (1918)
63. D.H. Hubel, T.N. Wiesel, Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. J. Physiol. **160**, 106–154 (1962)
64. J.B. Hutchinson, M.R. Uncapher, K.S. Weiner, D.W. Bressler, M.A. Silver et al., Functional heterogeneity in posterior parietal cortex across attention and episodic memory retrieval. Cereb. Cortex **24**, 49–66 (2012)
65. A. Ishai, L.G. Ungerleider, A. Martin, J.V. Haxby, The representation of objects in the human occipital and temporal cortex. J. Cogn. Neurosci. **12**(Suppl 2), 35–51 (2000)
66. C. Jacques, N. Witthoft, K.S. Weiner, B.L. Foster, V. Rangarajan et al., Corresponding ECoG and fMRI category-selective signals in human ventral temporal cortex. Neuropsychologia **83**, 14–28 (2016)
67. X. Jiang, E. Rosen, T. Zeffiro, J. Vanmeter, V. Blanz, M. Riesenhuber, Evaluation of a shape-based model of human face discrimination using FMRI and behavioral techniques. Neuron **50**, 159–172 (2006)
68. J. Jonas, S. Frismand, J.P. Vignal, S. Colnat-Coulbois, L. Koessler et al., Right hemispheric dominance of visual phenomena evoked by intracerebral stimulation of the human visual cortex. Hum. Brain Mapp. **35**, 3360–3371 (2014)
69. J. Jonas, B. Rossion, J. Krieg, L. Koessler, S. Colnat-Coulbois et al., Intracerebral electrical stimulation of a face-selective area in the right inferior occipital cortex impairs individual face discrimination. Neuroimage **99**, 487–497 (2014)
70. J. Jonas, C. Jacques, J. Liu-Shuang, H. Brissart, S. Colnat-Coulbois et al., A face-selective ventral occipito-temporal map of the human brain with intracerebral potentials. Proc. Natl. Acad. Sci. USA **113**, E4088–E4097 (2016)
71. N. Kanwisher, Domain specificity in face perception. Nat. Neurosci. **3**, 759–763 (2000)
72. N. Kanwisher, J. McDermott, M.M. Chun, The fusiform face area: a module in human extrastriate cortex specialized for face perception. J. Neurosci. **17**, 4302–4311 (1997)
73. N. Kanwisher, F. Tong, K. Nakayama, The effect of face inversion on the human fusiform face area. Cognition **68**, B1–B11 (1998)
74. K.N. Kay, J.D. Yeatman, Bottom-up and top-down computations in high-level visual cortex. Elife. 2017 Feb 22;6. pii: e22341 (2017)
75. K.N. Kay, T. Naselaris, R.J. Prenger, J.L. Gallant, Identifying natural images from human brain activity. Nature **452**, 352–355 (2008)
76. K.N. Kay, J. Winawer, A. Mezer, B.A. Wandell, Compressive spatial summation in human visual cortex. J. Neurophysiol. **110**, 481–494 (2013)
77. K.N. Kay, K.S. Weiner, K. Grill-Spector, Attention reduces spatial uncertainty in human ventral temporal cortex. Curr. Biol. **25**, 595–600 (2015)
78. S.M. Khaligh-Razavi, N. Kriegeskorte, Deep supervised, but not unsupervised, models may explain IT cortical representation. PLoS Comput. Biol. **10**, e1003915 (2014)
79. T.C. Kietzmann, J.D. Swisher, P. Konig, F. Tong, Prevalence of selectivity for mirror-symmetric views of faces in the ventral and dorsal visual pathways. J. Neurosci. **32**, 11763–11772 (2012)
80. M. Kim, M. Ducros, T. Carlson, I. Ronen, S. He et al., Anatomical correlates of the functional organization in the human occipitotemporal cortex. Magn. Reson. Imaging **24**, 583–590 (2006)

81. B.P. Klein, B.M. Harvey, S.O. Dumoulin, Attraction of position preference by spatial attention throughout human visual cortex. Neuron **84**, 227–237 (2014)
82. J. Konorski, *Integrative Activity of the Brain. An Interdisciplinary Approach* (The University of Chicago Press, Chicago, 1967)
83. N. Kriegeskorte, Deep neural networks: a new framework for modeling biological vision and brain information processing. Annu. Rev. Vis. Sci. **1**, 417–446 (2015)
84. N. Kriegeskorte, E. Formisano, B. Sorger, R. Goebel, Individual faces elicit distinct response patterns in human anterior temporal cortex. Proc. Natl. Acad. Sci. USA **104**, 20600–20605 (2007)
85. A. Krizhevsky, I. Sutskever, G. Hinton, Imagenet classification with deep convolutional neural networks. Presented at Neural Information Processing Systems (NIPS) (2012)
86. J. Kubilius, S. Bracci, H.P. Op de Beeck, Deep neural networks as a computational model for human shape sensitivity. PLoS Comput. Biol. **12**, e1004896 (2016)
87. Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard et al., Neural Information Processing (1989)
88. Y. Lee, B. Duchaine, H.R. Wilson, K. Nakayama, Three cases of developmental prosopagnosia from one family: detailed neuropsychological and psychophysical investigation of face processing. Cortex **46**, 949–964 (2010)
89. I. Levy, U. Hasson, G. Avidan, T. Hendler, R. Malach, Center-periphery organization of human object areas. Nat. Neurosci. **4**, 533–539 (2001)
90. G. Loffler, G. Yourganov, F. Wilkinson, H.R. Wilson, fMRI evidence for the neural representation of faces. Nat. Neurosci. **8**, 1386–1390 (2005)
91. G.R. Loftus, E.M. Harley, Why is it easier to identify someone close than far away? Psychon. Bull. Rev. **12**, 43–65 (2005)
92. S. Lorenz, K.S. Weiner, J. Caspers, H. Mohlberg, A. Schleicher et al., Two new cytoarchitectonic areas on the human mid-fusiform gyrus. Cereb. Cortex (2015)
93. A. Martin, C.L. Wiggs, L.G. Ungerleider, J.V. Haxby, Neural correlates of category-specific knowledge. Nature **379**, 649–652 (1996)
94. G. McCarthy, A. Puce, J.C. Gore, T. Allison, Face-specific processing in the human fusiform gyrus. J. Cogn. Neurosci. **9**, 605–610 (1997)
95. G. McCarthy, A. Puce, A. Belger, T. Allison, Electrophysiological studies of human face perception. II: response properties of face-specific potentials generated in occipitotemporal cortex. Cereb. Cortex **9**, 431–444 (1999)
96. E. McKone, Holistic processing for faces operates over a wide range of sizes but is strongest at identification rather than conversational distances. Vis. Res. **49**, 268–283 (2009)
97. K. Moutoussis, S. Zeki, The relationship between cortical activation and perception investigated with invisible stimuli. Proc. Natl. Acad. Sci. USA **99**, 9527–9532 (2002)
98. M. Mur, D.A. Ruff, J. Bodurka, P. De Weerd, P.A. Bandettini, N. Kriegeskorte, Categorical, yet graded-single-image activation profiles of human category-selective cortical regions. J. Neurosci. **32**, 8649–8662 (2012)
99. S. Nasr, N. Liu, K.J. Devaney, X. Yue, R. Rajimehr et al., Scene-selective cortical regions in human and nonhuman primates. J. Neurosci. **31**, 13771–13785 (2011)
100. V. Natu, M.A. Barnett, T. Hartley, J. Gomez, A. Stigliani, K. Grill-Spector, Development of neural sensitivity to face identity correlates with perceptual discriminability. J. Neurosci. **36**, 10893–10907 (2016)
101. V.S. Natu, A.J. O'Toole, Spatiotemporal changes in neural response patterns to faces varying in visual familiarity. Neuroimage **108**, 151–159 (2015)
102. V.S. Natu, F. Jiang, A. Narvekar, S. Keshvari, V. Blanz, A.J. O'Toole, Dissociable neural patterns of facial identity across changes in viewpoint. J. Cogn. Neurosci. **22**, 1570–1582 (2010)
103. A. Nestor, D.C. Plaut, M. Behrmann, Unraveling the distributed neural code of facial identity through spatiotemporal pattern analysis. Proc. Natl. Acad. Sci. USA **108**, 9998–10003 (2011)
104. K.M. O'Craven, P.E. Downing, N. Kanwisher, fMRI evidence for objects as the units of attentional selection. Nature **401**, 584–587 (1999)

105. J. Parvizi, C. Jacques, B.L. Foster, N. Withoft, V. Rangarajan et al., Electrical stimulation of human fusiform face-selective regions distorts face perception. J. Neurosci. **32**, 14915–14920 (2012)
106. M.V. Peelen, P.E. Downing, Within-subject reproducibility of category-specific visual activation with functional MRI. Hum. Brain Mapp. **25**, 402–408 (2005)
107. K.A. Pelphrey, N.J. Sasson, J.S. Reznick, G. Paul, B.D. Goldman, J. Piven, Visual scanning of faces in autism. J Autism Dev. Disord. **32**, 249–261 (2002)
108. M.A. Pinsk, M. Arcaro, K.S. Weiner, J.F. Kalkus, S.J. Inati et al., Neural representations of faces and body parts in macaque and human cortex: a comparative FMRI study. J. Neurophysiol. **101**, 2581–2600 (2009)
109. D. Pitcher, V. Walsh, G. Yovel, B. Duchaine, TMS evidence for the involvement of the right occipital face area in early face processing. Curr. Biol. **17**, 1568–1573 (2007)
110. D. Pitcher, D.D. Dilks, R.R. Saxe, C. Triantafyllou, N. Kanwisher, Differential selectivity for dynamic versus static information in face-selective cortical regions. Neuroimage **56**, 2356–2363 (2011)
111. D. Pitcher, V. Walsh, B. Duchaine, The role of the occipital face area in the cortical face perception network. Exp. Brain Res. **209**, 481–493 (2011)
112. D. Pitcher, T. Goldhaber, B. Duchaine, V. Walsh, N. Kanwisher, Two critical and functionally distinct stages of face and body perception. J. Neurosci. **32**, 15877–15885 (2012)
113. E. Privman, Y. Nir, U. Kramer, S. Kipervasser, F. Andelman et al., Enhanced category tuning revealed by intracranial electroencephalograms in high-order human visual areas. J. Neurosci. **27**, 6234–6242 (2007)
114. A. Puce, T. Allison, J.C. Gore, G. McCarthy, Face-sensitive regions in human extrastriate cortex studied by functional MRI. J. Neurophysiol. **74**, 1192–1199 (1995)
115. A. Puce, T. Allison, S. Bentin, J.C. Gore, G. McCarthy, Temporal cortex activation in humans viewing eye and mouth movements. J. Neurosci. **18**, 2188–2199 (1998)
116. A. Puce, T. Allison, G. McCarthy, Electrophysiological studies of human face perception. III: effects of top-down processing on face-specific potentials. Cereb. Cortex **9**, 445–458 (1999)
117. J.A. Pyles, T.D. Verstynen, W. Schneider, M.J. Tarr, Explicating the face perception network with white matter connectivity. PLoS One **8**, e61611 (2013)
118. R. Rajimehr, J.C. Young, R.B. Tootell, An anterior temporal face patch in human cortex, predicted by macaque maps. Proc. Natl. Acad. Sci. USA **106**, 1995–2000 (2009)
119. V. Rangarajan, D. Hermes, B.L. Foster, K.S. Weiner, C. Jacques et al., Electrical stimulation of the left and right human fusiform gyrus causes different effects in conscious face perception. J. Neurosci. **34**, 12828–12836 (2014)
120. J.H. Reynolds, D.J. Heeger, The normalization model of attention. Neuron **61**, 168–185 (2009)
121. M. Riesenhuber, T. Poggio, Hierarchical models of object recognition in cortex. Nat. Neurosci. **2**, 1019–1025 (1999)
122. M. Rosenke, K.S. Weiner, M.A. Barnett, K. Zilles, K. Amunts, R. Goebel, K. Grill-Spector, A cross-validated cytoarchitectonic atlas of the human ventral visual stream. NeuroImage **pii: S1053–8119**(17), 30151–30159 (2017)
123. B. Rossion, Constraining the cortical face network by neuroimaging studies of acquired prosopagnosia. Neuroimage **40**, 423–426 (2008)
124. B. Rossion, A. Boremanse, Robust sensitivity to facial identity in the right human occipito-temporal cortex as revealed by steady-state visual-evoked potentials. J. Vis. **11**, 1–18 (2011)
125. B. Rossion, R. Caldara, M. Seghier, A.M. Schuller, F. Lazeyras, E. Mayer, A network of occipito-temporal face-sensitive areas besides the right middle fusiform gyrus is necessary for normal face processing. Brain **126**, 2381–2395 (2003)
126. P. Rotshtein, R.N. Henson, A. Treves, J. Driver, R.J. Dolan, Morphing Marilyn into Maggie dissociates physical and identity face representations in the brain. Nat. Neurosci. **8**, 107–113 (2005)
127. D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, *Cognitive Modeling* (1988)

128. Z.M. Saygin, D.E. Osher, K. Koldewyn, G. Reynolds, J.D. Gabrieli, R.R. Saxe, Anatomical connectivity patterns predict face selectivity in the fusiform gyrus. Nat. Neurosci. **15**, 321–327 (2012)
129. C. Schiltz, B. Sorger, R. Caldara, F. Ahmed, E. Mayer et al., Impaired face discrimination in acquired prosopagnosia is associated with abnormal response to individual faces in the right middle fusiform gyrus. Cereb. Cortex **16**, 574–586 (2006)
130. C. Schiltz, L. Dricot, R. Goebel, B. Rossion, Holistic perception of individual faces in the right middle fusiform gyrus as evidenced by the composite face illusion. J. Vis. **10**(25), 1–16 (2010)
131. F. Schroff, D. Kalenichenko, J. Philbin, FaceNet: a unified embedding for face recognition and clustering. in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2015)* (2015), pp. 815–823
132. R.F. Schwarzlose, J.D. Swisher, S. Dang, N. Kanwisher, The distribution of category and location information across object-selective regions in human visual cortex. Proc. Natl. Acad. Sci. USA **105**, 4447–4452 (2008)
133. M.I. Sereno, A.M. Dale, J.B. Reppas, K.K. Kwong, J.W. Belliveau et al., Borders of multiple visual areas in humans revealed by functional magnetic resonance imaging. Science **268**, 889–893 (1995)
134. J. Sergent, J.L. Signoret, Functional and anatomical decomposition of face processing: evidence from prosopagnosia and PET study of normal subjects. Philos. Trans. R. Soc. Lond. B Biol. Sci. **335**, 55–61; discussion 61-2
135. J. Sergent, S. Ohta, B. MacDonald, Functional neuroanatomy of face and object processing. A positron emission tomography study. Brain **115**(Pt 1), 15–36 (1992)
136. T. Serre, A. Oliva, T. Poggio, A feedforward architecture accounts for rapid categorization. Proc. Natl. Acad. Sci. USA **104**, 6424–6429 (2007)
137. M.A. Silver, S. Kastner, Topographic maps in human frontal and parietal cortex. Trends Cogn. Sci. **13**, 488–495 (2009)
138. M.A. Silver, D. Ress, D.J. Heeger, Topographic maps of visual spatial attention in human parietal cortex. J. Neurophysiol. **94**, 1358–1371 (2005)
139. H.P. Snippe, J.J. Koenderink, Information in channel-coded systems: correlated receivers. Biol. Cybern. **67**, 183–190 (1992)
140. B. Sorger, R. Goebel, C. Schiltz, B. Rossion, Understanding the functional neuroanatomy of acquired prosopagnosia. Neuroimage **35**, 836–852 (2007)
141. T.C. Sprague, J.T. Serences, Attention modulates spatial priority maps in the human occipital, parietal and frontal cortices. Nat. Neurosci. **16**, 1879–1887 (2013)
142. J.K. Steeves, J.C. Culham, B.C. Duchaine, C.C. Pratesi, K.F. Valyear et al., The fusiform face area is not sufficient for face recognition: evidence from a patient with dense prosopagnosia and no occipital face area. Neuropsychologia **44**, 594–609 (2006)
143. C. Summerfield, E.H. Trittschuh, J.M. Monti, M.M. Mesulam, T. Egner, Neural repetition suppression reflects fulfilled perceptual expectations. Nat. Neurosci. (2008)
144. J.D. Swisher, M.A. Halko, L.B. Merabet, S.A. McMains, D.C. Somers, Visual topography of human intraparietal sulcus. J. Neurosci. **27**, 5326–5337 (2007)
145. Y. Taigman, M. Yang, M. Ranzato, L. Wolf, in *The IEEE Conference on Computer Vision and Pattern Recognition* (CVPR, 2014), pp. 1701–1708
146. H. Takemura, A. Rokem, J. Winawer, J.D. Yeatman, B.A. Wandell, F. Pestilli, A major human white matter pathway between dorsal and ventral visual cortex. Cereb. Cortex **26**, 2205–2214 (2016)
147. T. Tallinen, J.Y. Chung, J.S. Biggins, L. Mahadevan, Gyrification from constrained cortical expansion. Proc. Natl. Acad. Sci. USA **111**, 12667–12672 (2014)
148. J.W. Tanaka, M.J. Farah, Parts and wholes in face recognition. Q. J. Exp. Psychol. A Hum. Exp. Psychol. **46**, 225–245 (1993)
149. I. Tavor, M. Yablonski, A. Mezer, S. Rom, Y. Assaf, G. Yovel, Separate parts of occipito-temporal white matter fibers are associated with recognition of faces and places. Neuroimage (2013)

150. F. Tong, K. Nakayama, J.T. Vaughan, N. Kanwisher, Binocular rivalry and visual awareness in human extrastriate cortex. Neuron **21**, 753–759 (1998)
151. F. Tong, K. Nakayama, M. Moscovitch, O. Weinrib, N. Kanwisher, Response properties of the human fusiform face area. Cogn. Neuropsychol. **17**, 257–280 (2000)
152. D. Tsao, M. Livingstone, Mechanisms of face perception. Annu. Rev. Neurosci. **31**, 411–437 (2008)
153. D.Y. Tsao, S. Moeller, W.A. Freiwald, Comparing face patch systems in macaques and humans. Proc. Natl. Acad. Sci. USA **105**, 19514–19519 (2008)
154. T. Valentine, Face-space models of face recognition, in *Computational, Geometric, and Process Perspectives on Facial Cognition: Contexts and Challenges*, ed. by M.J. Wenger, J.T. Townsend (Lawrence Erlbaum Associates Inc, Hillsdale, 2001)
155. G. Van Belle, P. De Graef, K. Verfaillie, T. Busigny, B. Rossion, Whole not hole: expert face recognition requires holistic perception. Neuropsychologia **48**, 2620–2629 (2010)
156. G. Van Belle, T. Busigny, P. Lefevre, S. Joubert, O. Felician et al., Impairment of holistic face perception following right occipito-temporal damage in prosopagnosia: converging evidence from gaze-contingency. Neuropsychologia **49**, 3145–3150 (2011)
157. D.C. Van Essen, J.L. Gallant, Neural mechanisms of form and motion processing in the primate visual system. Neuron **13**, 1–10 (1994)
158. D.C. Van Essen, C.H. Anderson, D.J. Felleman, Information processing in the primate visual system: an integrated systems perspective. Science **255**, 419–423 (1992)
159. P. Vuilleumier, R.N. Henson, J. Driver, R.J. Dolan, Multiple levels of visual object constancy revealed by event-related fMRI of repetition priming. Nat. Neurosci. **5**, 491–499 (2002)
160. P. Vuilleumier, J.L. Armony, J. Driver, R.J. Dolan, Distinct spatial frequency sensitivities for processing faces and emotional expressions. Nat. Neurosci. **6**, 624–631 (2003)
161. B.A. Wandell, J. Winawer, Imaging retinotopic maps in the human brain. Vis. Res. **51**, 718–737 (2011)
162. B.A. Wandell, J. Winawer, Computational neuroimaging and population receptive fields. Trends Cogn. Sci. **19**, 349–357 (2015)
163. B.A. Wandell, S.O. Dumoulin, A.A. Brewer, Visual field maps in human cortex. Neuron **56**, 366–383 (2007)
164. K. Weibert, T.J. Andrews, Activity in the right fusiform face area predicts the behavioural advantage for the perception of familiar faces. Neuropsychologia **75**, 588–596 (2015)
165. K.S. Weiner, K. Grill-Spector, Sparsely-distributed organization of face and limb activations in human ventral temporal cortex. Neuroimage **52**, 1559–1573 (2010)
166. K.S. Weiner, K. Grill-Spector, The improbable simplicity of the fusiform face area. Trends Cogn. Sci. **16**(5), 251–4 (2012)
167. K.S. Weiner, K. Grill-Spector, The evolution of face processing networks. Trends Cogn. Sci. **19**, 240–241 (2015)
168. K.S. Weiner, K. Zilles, The anatomical and functional specialization of the fusiform gyrus. Neuropsychologia **83**, 48–62 (2016)
169. K.S. Weiner, R. Sayres, J. Vinberg, K. Grill-Spector, fMRI-adaptation and category selectivity in human ventral temporal cortex: regional differences across time scales. J. Neurophysiol. **103**, 3349–3365 (2010)
170. K.S. Weiner, G. Golarai, J. Caspers, M.R. Chuapoco, H. Mohlberg et al., The mid-fusiform sulcus: a landmark identifying both cytoarchitectonic and functional divisions of human ventral temporal cortex. Neuroimage **84**, 453–465 (2014)
171. K.S. Weiner, J. Jonas, J. Gomez, L. Maillard, H. Brissart et al., The face-processing network is resilient to focal resection of human visual cortex. J. Neurosci. **36**, 8425–8440 (2016)
172. K.S. Weiner, J.D. Yeatman, B.A. Wandell, The posterior arcuate fasciculus and the vertical occipital fasciculus. Cortex. 31 Mar 2016. pii: (16)30050-8 (2016). doi:10.1016/S0010-9452
173. K.S. Weiner, B.A. Barnett, S. Lorenz, J. Caspers, A. Stigliani et al., The cytoarchitecture of domain-specific regions in human high-level visual cortex. Cereb. Cortex (2017)
174. Y. Weiss, S. Edelman, M. Fahle, Models of perceptual learning in vernier hyperacuity. Neural Comput. **5**, 695–718 (1993)

175. J.S. Winston, R.N. Henson, M.R. Fine-Goulden, R.J. Dolan, fMRI-adaptation reveals dissociable neural representations of identity and expression in face perception. J. Neurophysiol. **92**, 1830–1839 (2004)
176. N. Witthoft, S. Poltoratski, M. Nguyen, G. Golarai, A. Liberman et al., Developmental prosopagnosia is associated with reduced spatial integration in the ventral visual cortex, in *bioRxiv* (2016)
177. D.L. Yamins, J.J. DiCarlo, Using goal-driven deep learning models to understand sensory cortex. Nat. Neurosci. **19**, 356–365 (2016)
178. D.L. Yamins, H. Hong, C.F. Cadieu, E.A. Solomon, D. Seibert, J.J. DiCarlo, Performance-optimized hierarchical models predict neural responses in higher visual cortex. Proc. Natl. Acad. Sci. USA **111**, 8619–8624 (2014)
179. J.D. Yeatman, K.S. Weiner, F. Pestilli, A. Rokem, A. Mezer, B.A. Wandell, The vertical occipital fasciculus: a century of controversy resolved by in vivo measurements. Proc. Natl. Acad. Sci. USA **111**, E5214–E5223 (2014)
180. D.J. Yi, T.A. Kelley, R. Marois, M.M. Chun, Attentional modulation of repetition attenuation is anatomically dissociable for scenes and faces. Brain Res. **1080**, 53–62 (2006)
181. G. Yovel, N. Kanwisher, Face perception: domain specific, not process specific. Neuron **44**, 889–898 (2004)
182. G. Yovel, N. Kanwisher, The neural basis of the behavioral face-inversion effect. Curr. Biol. **15**, 2256–2262 (2005)
183. X. Yue, B.S. Cassidy, K.J. Devaney, D.J. Holt, R.B. Tootell, Lower-level stimulus features strongly influence responses in the fusiform face area. Cereb. Cortex **21**, 35–47 (2011)
184. S. Zeki, S. Shipp, The functional logic of cortical connections. Nature **335**, 311–317 (1988)

Chapter 2
Real-Time Face Identification via Multi-convolutional Neural Network and Boosted Hashing Forest

Yury Vizilter, Vladimir Gorbatsevich, Andrey Vorotnikov and Nikita Kostromov

Abstract The family of real-time face representations is obtained via Convolutional Network with Hashing Forest (CNHF). We learn the CNN, then transform CNN to the multiple convolution architecture and finally learn the output hashing transform via new Boosted Hashing Forest (BHF) technique. This BHF generalizes the Boosted Similarity Sensitive Coding (SSC) approach for hashing learning with joint optimization of face verification and identification. CNHF is trained on CASIA-WebFace dataset and evaluated on LFW dataset. We code the output of single CNN with 97% on LFW. For Hamming embedding we get CBHF-200 bit (25 byte) code with 96.3% and 2,000-bit code with 98.14% on LFW. CNHF with 2,000×7-bit hashing trees achieves 93% rank-1 on LFW relative to basic CNN 89.9% rank-1. CNHF generates templates at the rate of 40+ fps with CPU Core i7 and 120+ fps with GPU GeForce GTX 650.

2.1 Introduction

Various face recognition applications presume different priorities of template size, template generation speed, template matching speed, and recognition rates. We know that the fastest search in a database is provided by binary templates with Hamming distance [1, 7–10, 12, 14, 18, 20, 21, 30, 34]. On the other hand, the best face recognition rates are achieved by deep convolutional neural networks (CNN) with non-binary face representations [3, 5, 23–25, 27, 29, 31, 35]. These approaches can

Y. Vizilter (✉) · V. Gorbatsevich · A. Vorotnikov · N. Kostromov
State Research Institute of Aviation Systems (GosNIIAS), Moscow, Russia
e-mail: viz@gosniias.ru

V. Gorbatsevich
e-mail: gvs@gosniias.ru

A. Vorotnikov
e-mail: vorotnikov@gosniias.ru

N. Kostromov
e-mail: nikita-kostromov@yandex.ru

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_2

be fused in the special CNN architecture with binary output layer, which we refer as Convolutional Network with Hashing Layer (CNHL). The most promising CNHL is described in [6], where CNN and hashing layer are learned together via back propagation technique. But now we need the family of face representations, which continuously varies from small Hamming codes to coded features with larger size, better metrics and higher recognition rates. So, in this chapter we propose to combine the CNN and additional hashing transform based on Hashing Forest (HF). Our HF forms the vector of features coded by binary trees. HF with different depth of trees and different coding objectives allows obtaining the family of face representations based on the same CNN. We refer such CNN+HF architecture as Convolutional Network with Hashing Forrest (CNHF). In case of 1-bit coding trees CNHF degrades to CNHL and provides the Hamming embedding.

The architecture of our CNHF is based on the Max-Feature-Map (MFM) CNN architecture proposed by Xiang Wu [31]. For real-time implementation we accelerate our CNN via transforming to the multiple convolution architecture.

We propose the new Boosted Hashing Forest (BHF) technique, which generalizes the Boosted Similarity Sensitive Coding (Boosted SSC) [20, 21] for discriminative data coding by forest hashing with direct optimization of objective function and given properties of coded feature space. We also introduce and implement the new biometric-specific objective function for joint optimization of face verification and identification.

Proposed CNHF face representations are trained on CASIA-WebFace dataset and evaluated on LFW dataset. Our experiments demonstrate both compact binary face representations and increasing of face verification and identification rates. In the Hamming embedding task BHF essentially outperforms the original Boosted SSC. Our CNHF 200 bit (25 byte) hash achieves 96.3% on LFW with 70-time gain in a matching speed. CNHF 2,000 bit hash provides 98.14% on LFW. CNHF with 2,000$\times$7-bit hashing trees achieves 93% rank-1 on LFW relative to basic CNN 89.9% rank-1.

The remainder of this chapter is organized as follows. Section 2.2 briefly describes the related work. Section 2.3 describes the architecture and learning of our CNHF with multiple convolutional layers. Section 2.4 contains the outline of proposed BHF technique and its implementation for face hashing. Experimental results are presented in Sect. 2.5. Conclusion and discussion are presented in Sect. 2.6.

2.2 Related Work

A lot of face representation techniques were proposed [4, 16, 26], but all state-of-the-art results are obtained now via deep CNN. One can learn CNN for multi-class face identification with classes corresponding to persons [27, 35], or learn the similarity metric by training two identical CNNs (Siamese Architecture [5, 29], or combine these approaches [23, 25, 32]). Best modern results on LFW are obtained by ensembles of deep nets learned on different parts (patches) of face [13, 23, 25].

Nevertheless, some single nets can be efficient enough with essentially lower computational cost [3, 31]. Most frequently the CNN-based face representation is formed as an output of top hidden layer [5, 23, 27, 29, 31, 35]. Sometimes the PCA is applied for size reduction [23, 24]. The L2-distance [4, 29] or cosine similarity [23, 27, 31] are of use for matching of face representations.

Binary hashing means the assigning of binary code to each input feature vector. The review of classical hashing techniques is presented in [9]. The simplest binary hashing idea is to use some dimensionality reduction transform and then apply some quantization technique. The optimization-based hashing approach presumes the similarity-driven data embedding into the Hamming space. In [7] the similarity search is proposed based on linear binary coders and vectors of weights obtained by random rotations. The Iterative Quantization (ITQ) technique [8] considers the hashing problem as a search of rotation, which minimizes the quantization error. Kernel-Based Supervised Hashing (KSH) [14] utilizes a kernel formulation for the target hash functions. The affinity-preserving algorithm [10] performs k-means clustering and learns the binary indices of the quantized cells. The manifold hashing techniques follow the ideas of manifold learning. The Spectral Hashing [30] relaxes the hashing problem in the manner of Laplacian Eigenmaps [1]. Topology Preserving Hashing (TPH) [34] performs the Hamming embedding with additional preserving the neighbor ranks. Locally Linear Hashing (LLH) [12] presumes both preserving distances and reconstructing the locally linear structures. The Semantic Hashing (SH) [18] solves the hashing problem with the use of Restricted Boltzmann Machines (RBM). Boosted Similarity Sensitive Coding (Boosted SSC) proposed by Shaknarovich, Voila and Darrell [20, 21] performs the sequential bit-by-bit growing of the hash code with reweighting of samples in the manner of AdaBoost and forming the weighted Hamming space.

The idea of binary face coding based on deep learning is well implemented in [6]. The CNN and hashing layer are learned together via back propagation technique, and 32-bit binary face representation is generated with 91% verification on LFW. Unfortunately, the direct optimization of more complex face coding criterions is not available in this one-step CNHL learning framework. In particular, it cannot provide the immediate optimization of Cumulative Matching Curve (CMC). Due to this we implement the two-step CNHF learning procedure: learning basic CNN first and hashing transform second.

Our hashing transform is based on hashing forest. Look at some previous forest hashing techniques. Qiu, Sapiro, and Bronstein [17] propose the random forest semantic hashing scheme with information-theoretic code aggregation for large-scale data retrieval. The feature induction based on random forest for learning regression and multi-label classification is proposed by Vens and Costa [28]. Yu and Yuan [33] implement a forest hashing with special order-sensitive Hamming distance. The forest hashing by Springer et al. [22] combines *kd*-trees with hashing technique. The Boosted Random Forest algorithm proposed by Mishina, Tsuchiya, and Fujiyoshi [15] is out of the binary hashing topic. Our approach performs the feature space coding via boosted forest hashing in the manner of Boosted SSC with optimizing of task-specific objective function. So, we mainly consider our BHF technique as a generalization of Boosted SSC.

2.3 CNHF with Multiple Convolution CNN

Our CNHF contains the basic deep CNN and additional hashing transform based on Hashing Forrest (HF). This hashing forest forms the output CNHF binary face representation, which semantically corresponds to some objective vector of features coded by these binary trees (Fig. 2.1). For obtaining the family of optimized face representations based on the same CNN we use the two-step CNHF learning procedure. At the first step the CNN is formed and trained for multi-class face identification. At the second step the hashing transform is trained for combined face verification and identification. We start from learning the source CNN with softmax output layer for face identification. Then we transform its convolutional layers to the multiple convolution form. Finally we cut the output softmax layer and use the activations of top hidden layer as a basic face representation for further hashing. In this chapter, we use the Max-Feature-Map (MFM) CNN architecture proposed by Xiang Wu [31]. It is based on the Max-Feature-Map activation function instead of ReLU. Reference [31] demonstrates that Max-Feature-Map can get the compact and discriminative feature vectors. The source network architecture contains four convolutional layers, four layers of pooling + MFM pooling, one fully connected layer and the softmax

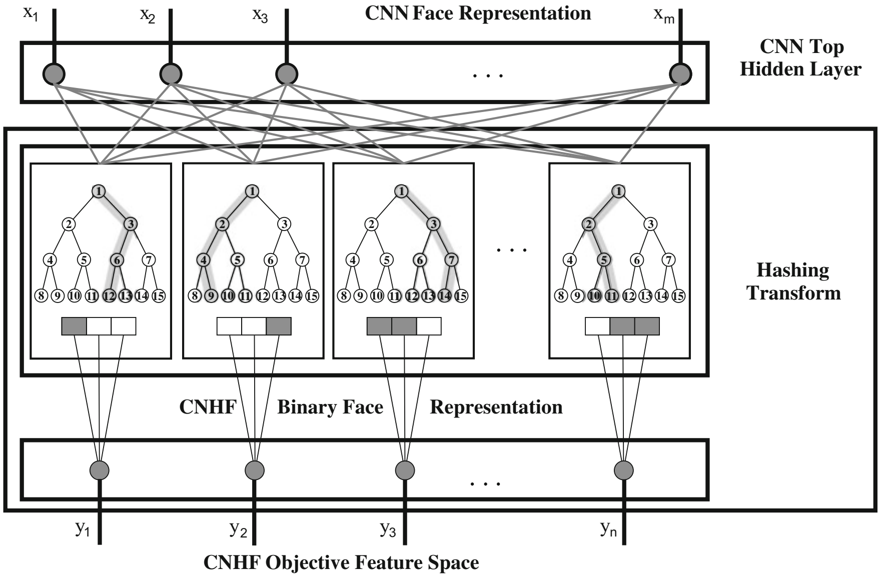


Fig. 2.1 Architecture of CNHF: CNN + hashing transform based on hashing forest

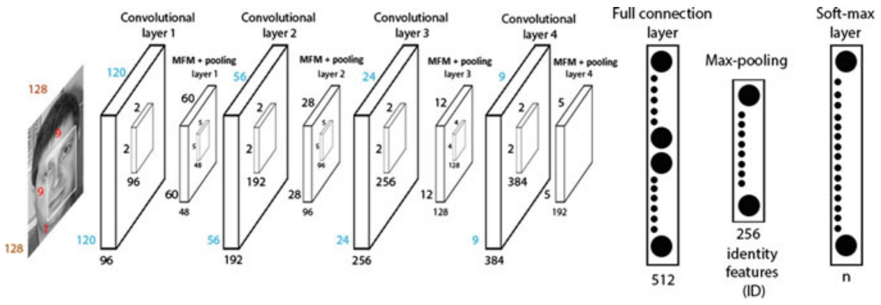


Fig. 2.2 Architecture of source MFM deep net [25]

layer (Fig. 2.2). Following the approach of Xiang Wu [31] we start from learning this source MFM deep net for multi-class face identification with classes corresponding to persons in the manner [24, 31] using the backpropagation technique.

Unfortunately, we cannot directly implement the architecture (Fig. 2.2) for the real-time face identification with CPU. We need to optimize this architecture in order to obtain essentially higher calculation speed. So, we propose and apply the new approach for sequential transformation of deep network topology based on the following tricks:

1. We use the small-sized filters instead of large-sized filters in the convolutional layers. For example, we substitute one layer with 5×5 filters by the sequence of two layers with 3×3 filters, which is 1.38 times faster on CPU.
2. We decrease the number of filters in each layer. For example, the first layer of source net (Fig. 2.2) contains 96 filters, but the first layer of our transformed net contains 20 filters only, which is more than 4 times faster on CPU.
3. The each layer is transformed and relearned separately. For this purpose we need to provide the equal input and output dimensionalities for the source layer and corresponding part of transformed net, which is used for its substitution. We do this by adding the $1 \times 1 \times n$ layers to the transformed net, where n is the number of filters in the substituted source layer. For example, we substitute the one $9 \times 9 \times 96$ layer of source network (Fig. 2.2) by the sequence of two layers $9 \times 9 \times 20$ and $1 \times 1 \times 96$, which is still more than 4 times faster on CPU.

Thus, we simplify the network topology sequentially, layer by layer, without the relearning of the whole CNN. In this process we represent the each convolution as a combination of convolutions, so, we refer the resultant architecture of transformed net as “multiple convolutional” or briefly “multiconv”. At the each step of one layer substitution we use the Euclidean loss for minimization of difference between the output response of this source layer and corresponding part of transformed net, which is used for its substitution, for the same input values. Figure 2.3 illustrates the proposed scheme for topology transformation and relearning. The training set at this stage contains face images without pointing to persons. We use the open source

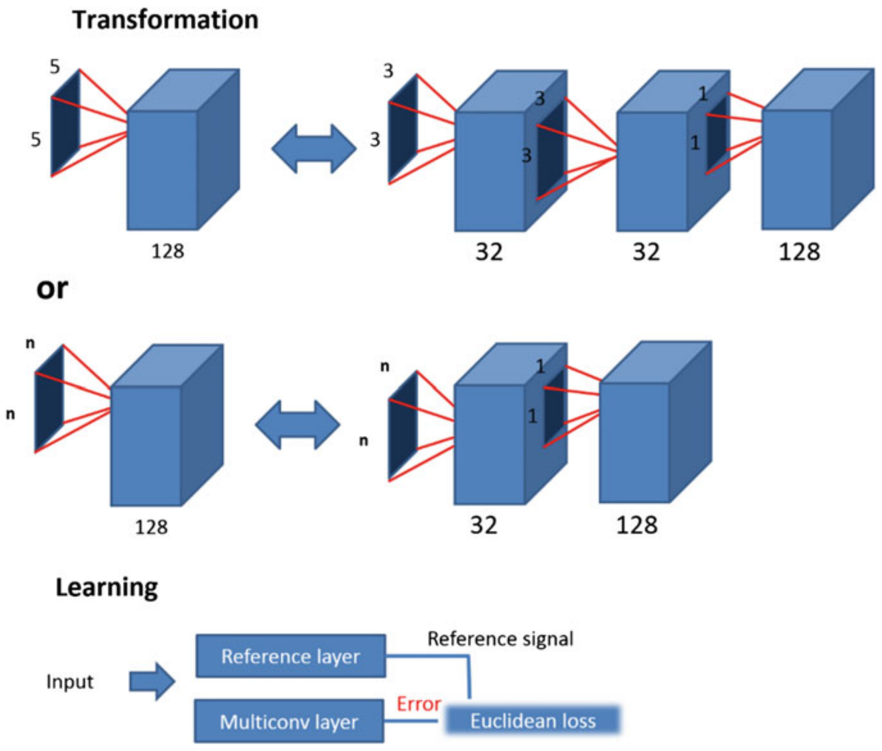


Fig. 2.3 The proposed scheme for network topology transformation and relearning

framework Caffe for learning of transformed layers by standard back propagation technique as well as for the whole network training (see Sect. 2.5.1).

Finally, we could represent the proposed process for deep net architecture sequential transformation as the following informal Algorithm 0.

```
Algorithm 0: CNN transform to multiconv net
Input data: conv net.
Output data: multiconv net.
Initialization:
    multiconv net := conv net.
Repeat iterations:
  Step 1. Find the slowest conv layer $L$ in multiconv net;
  Step 2. Replace $L$ by sequence of layers $S$ with less summarized number of convolutions;
  Step 3. Learn $S$ using the Euclidean loss for imitating the output of substituted layer $L$;
while speed grows and accuracy is still high enough
```

Table 2.1 Iteration 1

Layer	Convolutional layer 2 (Fig. 2.2)	Convolutional layer 3 (Fig. 2.4)	Convolutional layer 4 (Fig. 2.4)	Convolutional layer 5 (Fig. 2.4)	Convolutional layer 6 (Fig. 2.4)
Filter size	5 × 5 × 48	3 × 3 × 48	1 × 1 × 24	3 × 3 × 32	1 × 1 × 32
Number of filters	192	24	32	32	192
Number of operations (mult.)	722,534,400	34,877,952	2,583,552	28,901,376	19,267,584

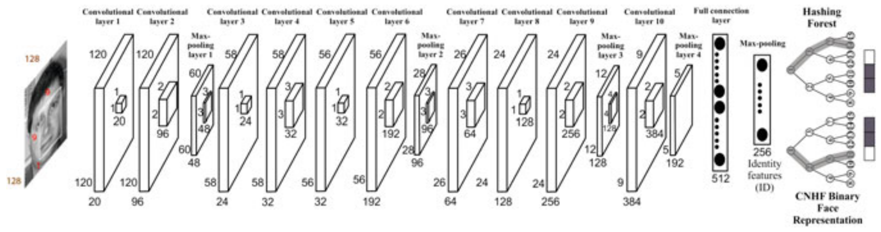


Fig. 2.4 Architecture of CNHF based on MFM net with multiple convolutions

Table 2.2 Iteration 2

Layer	Convolutional layer 3 (Fig. 2.2)	Convolutional layer 7 (Fig. 2.4)	Convolutional layer 8 (Fig. 2.4)	Convolutional layer 9 (Fig. 2.4)
Filter size	5 × 5 × 96	3 × 3 × 96	3 × 3 × 64	1 × 1 × 128
Number of filters	256	64	128	256
Number of operations (mult.)	353,894,400	37,380,096	42,467,328	18,874,368

Actually, only three iterations of the Algorithm 0 were used.

Iteration 1: Convolutional layer 2 (Fig. 2.2) was replaced by layers 3, 4, 5, 6 (see the Table 2.1):

The original layer (layer 2 in Fig. 2.2) requires more than 700 million multiplications and produces the output with the dimensions of 56 × 56 × 192, the layer sequence (layers 3, 4, 5, 6 in Fig. 2.4) requires about 90 million of multiplications (about 7 times less than the original layer) with the same output size.

Iteration 2: Convolutional layer 3 (Fig. 2.2) was replaced by layers 7, 8, 9 (Table 2.2):

The original layer (layer 3 in Fig. 2.2) requires more than 350 million multiplications and produces the output with the dimensions of 24 × 24 × 256, the layer sequence (layers 7, 8, 9 in Fig. 2.4) requires about 100 million multiplications (about 3 times less than the original layer) with the same output size.

Table 2.3 Iteration 3

Layer	Convolutional layer 1 (Fig. 2.2)	Convolutional layer 1 (Fig. 2.4)	Convolutional layer 2 (Fig. 2.4)
Filter size	$9 \times 9 \times 1$	$9 \times 9 \times 1$	$1 \times 1 \times 20$
Number of filters	96	20	96
Number of operations (mult.)	111,974,400	23,328,000	27,648,000

Iteration 3: Convolutional layer 1 (Fig. 2.2) was replaced by layers 1, 2 (Table 2.3):
The original layer (layer 1 in Fig. 2.2) requires more than 110 million multiplications and produces the output with the dimensions of $120 \times 120 \times 96$, the layer sequence (layers 1, 2 in Fig. 2.4) requires about 50 million multiplications (about 2 times less than the original layer) with the same output size.

Unfortunately, all our attempts to replace layer 4 from the original network led to significant loss in accuracy (greater than 10%), but this layer requires relatively less computations – about 90 million multiplications.

The main advantage of this approach is a relatively high speed of learning at the steps of layer substitutions. We used the GTX 1080 card for this learning and trained the one multiconv substitution approximately in 10–15 min. This allows performing the multiconv transformation of any source CNN architecture in the very convenient and partially automated way.

After all simplifying substitutions, the transformed CNN is trained again for multi-class face identification with classes corresponding to persons in the manner [24, 31] using the backpropagation technique. Finally the output softmax layer of transformed MFM net is replaced by hashing forest, and we obtain the CNHF based on MFM with multiple convolutional layers (Fig. 2.4). In result our CNHF contains 10 convolutional layers, four layers of MFM+pooling, fully connected layer and hashing forest. This CNHF generates face templates at the rate of 40+ fps with CPU Core i7 and 120+ fps with GPU GeForce GTX 650. Thus, we can conclude that the proposed multiconv approach makes our CNHF 5 times faster on CPU than source CNN. It is enough for the real-time operation.

2.4 Learning Face Representation via Boosted Hashing Forest

2.4.1 Boosted SSC, Forest Hashing and Boosted Hashing Forest

We learn our hashing transform via the new Boosted Hashing Forest (BHF) technique, which combines the algorithmic structure of Boosted SSC [20, 21] and the binary code structure of forest hashing [15, 17, 22, 28, 33].

Boosted SSC algorithms optimize the performance of L1 distance in the embedding space as a proxy for the pairwise similarity function, which is conveyed by a set of examples of positive (similar) and negative (dissimilar) pairs. The *SSC algorithm* takes pairs labeled by similarity and produces a binary embedding space. The embedding is learned by independent collecting thresholded projections of the input data. The threshold is selected by optimal splitting the projections of negative pairs and non-splitting the projections of positive pairs. *Boosted SSC algorithm* collects the embedding dimensions greedily with adaptive weighting of samples and dimensions in the manner of AdaBoost. *BoostPro algorithm* uses a soft thresholding for gradient-based learning of projections.

The differences of proposed BHF w.r.t. Boosted SSC are the following:

1. BHF performs the binary coding of output feature space, which is not binary in general, but can be binary Hamming, if required.
2. BHF performs the direct optimization of any given objective function of output features.
3. BHF learns the objective-driven data projections via RANSAC algorithm without gradient-based optimization.
4. BHF performs the recursive coding by binary trees and forms the hashing forest, while Boosted SSC performs the iterative feature coding and forms hashing vector.
5. BHF performs the adaptive reweighting of training pairs based on their contribution to the objective function, unlike the AdaBoost-style reweighting of Boosted SSC.
6. Boosted SSC forms the weighted Hamming space. Our BHF forms the any given metric space, including non-weighted Hamming space for fastest data search.

Algorithm 1: Greedy ORC

```
Input data: $X$, $\mathcal{J}$, $n_{ORC}$.
Output data: $\mathbf{h}(\mathbf{x})$: $\mathbf{x}\in R^m \to \mathbf{y}\in\{0,1\}^{n_{ORC}}$, $\mathbf{h}(\mathbf{x})\in\mathbf{H}$.
Initialization:
  Step 0. $k$:=0; $\mathbf{h}^{(k)}$ := ( ).
Repeat iterations:
  $k$:= $k$+1;
  Learn $k$-th elementary coder:
    $h^{(k)}(\mathbf{x}, \mathbf{h}^{(k-1)})$:= Learn1BitHash($\mathcal{J}$, $X$, $\mathbf{h}^{(k-1)}$);
  Add $k$-th elementary coder to the hashing function:
    $\mathbf{h}^{(k)}(\mathbf{x})$ := ($\mathbf{h}^{(k-1)}(\mathbf{x})$, $h^{(k)}(\mathbf{x}, \mathbf{h}^{(k-1)})$);
while $k<n_{ORC}$. // stop if the given size of coder is got
```

The main differences of proposed BHF w.r.t. other forest hashing techniques: we obtain the hashing forest via RANSAC projections and boosting process in the manner of Boosted SSC; we optimize the task-specific objective function in the coded feature space, but not the similarity in the binary code space.

BHF implementation for face recognition has some additional original features: new biometric-specific objective function with joint optimization of face verification and identification; selection and processing of subvectors of the input feature vector;

creation of ensemble of independent hash codes for overcoming the limitations of greedy learning. In the next subsections we describe our BHF algorithms in detail.

2.4.2 BHF: Objective-Driven Recurrent Coding

Let the training set $X = \{\mathbf{x}_i \in R^m\}_{i=1,\ldots,N}$ contains N objects described by m-dimensional feature vectors. Map X to the n-dimensional binary space: $X = \{\mathbf{x}_i \in R^m\}_{i=1,\ldots,N} \rightarrow B = \{\mathbf{b}_i \in \{0, 1\}^n\}_{i=1,\ldots,N}$. This mapping is an *n-bit coder*:

$$\mathbf{h}(\mathbf{x}) : \mathbf{x} \in R^m \rightarrow \mathbf{b} \in \{0, 1\}^n \tag{2.1}$$

The elementary coder is called the 1-*bit hashing function*

$$h(\mathbf{x}) : \mathbf{x} \in R^m \rightarrow \quad b \in \{0, 1\} \tag{2.2}$$

Let some *objective function* (coding criterion) is given and required to be minimized

$$\mathcal{J}(X, \mathbf{h}) \rightarrow min(\mathbf{h}). \tag{2.3}$$

```
Algorithm 2: RANSAC Learn1ProjectionHash
Input data: $\mathcal{J}$, $X$, $\mathbf{h}^{(k-1)}$, $k_{RANSAC}$.
Output data: $h(\mathbf{w}, t, \mathbf{x})$.
Initialization:
    Step 0. $k$:=0; $\mathcal{J}_{min}$:=+∞.
Repeat iterations:
    $k$:= $k$+1;
    Step 1. Take the random dissimilar pair ($\mathbf{x}_i$ ,$\mathbf{x}_j$) in $X$.
    Step 2. Get vector $\overrightarrow{(\mathbf{x}_i, \mathbf{x}_j)}$ as a vector of hyperplane
  direction: $\mathbf{w}_k := \mathbf{x}_j - \mathbf{x}_i$.
    Step 3. Calculate the threshold $t_k$ minimizing $\mathcal{J}$ (6) by
  $t$ with $\mathbf{w}=\mathbf{w}_k$: $t_k := \mathrm{argmin}_t\ \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t)$.
    Step 4. If $\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t_k) < \mathcal{J}_{min}$, then
            $\mathcal{J}_{min} := \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t_k)$; $\mathbf{w} := \mathbf{w}_k$; $t := t_k$.
while $k<k_{RANSAC}$. // stop if the given number of RANSAC
iterations is achieved
```

Denote $\mathbf{h}^{(k)}(\mathbf{x}) = (h^{(1)}(\mathbf{x}),\ldots,h^{(k)}(\mathbf{x}))$. The operation of coders concatenation is $\mathbf{h}^{(k)}(\mathbf{x}) := (\mathbf{h}^{(k-1)}(\mathbf{x}),\ h^{(k)}(\mathbf{x}))$. The Greedy Objective-driven Recurrent Coding (Greedy ORC) algorithm (Algorithm 1) sequentially forms the bits of our coder in a recurrent manner: $h^{(k)}(\mathbf{x}) = h^{(k)}(\mathbf{x},\mathbf{h}^{(k-1)})$. The proper procedure for learning the each kth bit is described in the next subsections.

2.4.3 BHF: Learning Elementary Projection via RANSAC Algorithm

At the kth step of coder growing

$$\mathcal{J}(X, \mathbf{h}^{(k)}) = \mathcal{J}(X, \mathbf{h}^{(k-1)}, h^{(k)}) \to min\{h^{(k)} \in \mathbf{H}\}, \tag{2.4}$$

Algorithm 3: Optimal threshold selection

```
Input data: $\mathcal{J}$, $X$, $\mathbf{h}^{(k-1)}$, $\mathbf{w}$, $N$.
Output data: $t$.
Operations:
    Step 1. For all $\mathbf{x}_i \in X$ calculate projections $t_i = (\mathbf{x}_i, \mathbf{w})$;
    Step 2. Arrange samples $\mathbf{x}_i$ and projections $t_i$ by increasing of $t_i$. For all arranged indices $i$=1..$N$ do:
        $\Delta\mathcal{J}_i := 0$;
    Step 3. For all pairs $(\mathbf{x}_i, \mathbf{x}_j)$ with $t_i < t_j$ do:
        // increment the step values at projection points
      $\Delta\mathcal{J}_1 := \Delta\mathcal{J}_1 + \mathcal{J}_{ij}^{(out)}$;
      $\Delta\mathcal{J}_i := \Delta\mathcal{J}_i + \mathcal{J}_{ij}^{(in)} - \mathcal{J}_{ij}^{(out)}$;
      $\Delta\mathcal{J}_j := \Delta\mathcal{J}_j + \mathcal{J}_{ij}^{(out)} - \mathcal{J}_{ij}^{(in)}$;
    Step 4. Recover the stepwise objective function and find the optimal threshold
        $\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t_1) := 0$; // recover the first value
        $i$:=0; $\mathcal{J}_{min}$:=+$\infty$.
        Repeat iterations:
          $i := i+1$;
          $\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t_i) := \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t_{i-1}) + \Delta\mathcal{J}_i$;
          // accumulate step values from left to right
          If $\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t_i) < \mathcal{J}_{min}$, then
                $\mathcal{J}_{min} := \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t_i)$; $t := (t_i + t_{i+1})/2$;
        while $i<N$-1. // stop if all step points are tested
```

where **H** is a class of coders. Consider the class of elementary coders based on thresholded linear projections

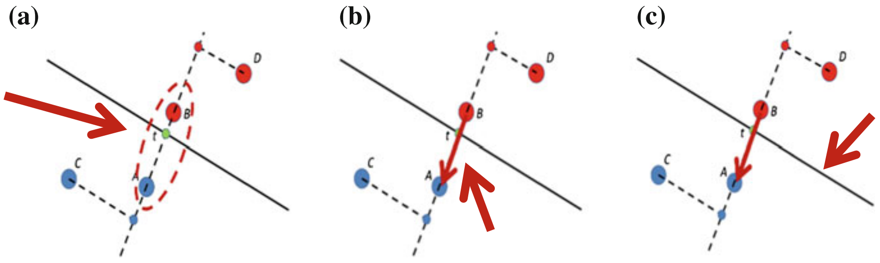


Fig. 2.5 RANSAC Learn1ProjectionHash: **a** *Step 1*, **b** *Steps 2*, **c** *Steps 3* of Algorithm 2

$$h(\mathbf{w}, t, \mathbf{x}) = sgn(\textstyle\sum_{k=1,\dots,m} w_k x_k + t), \tag{2.5}$$

where $\mathbf{w}$ – vector of weights, t – threshold of hashing function, $sgn(u) = \{1$, if $u > 0$; 0 - otherwise$\}$. In case of (2.5) function (2.4) takes the form

$$\mathcal{J}(X, \mathbf{h}^{(k-1)}, h^{(k)}) = \mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t) \rightarrow min\{\mathbf{w} \in R^m, t \in R\}. \tag{2.6}$$

We use the RANSAC algorithm for approximate solving (2.6). RANSAC hypotheses about $\mathbf{w}$ parameters are generated based on the random choice of dissimilar pairs in a training set (Algorithm 2, Fig. 2.5).

In general case the determination of optimal threshold at the step 3 of this Algorithm 2 could require a lot of time. But in important particular case, the objective function can be represented as a sum of some values corresponded to all pairs of samples from the training set. If these values for each pair depend only on the fact, whether the threshold separates the projections of these samples, or not, then the objective function will be a stepwise function

$$\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}, t) = \textstyle\sum_{i=1,\dots,N} \sum_{j=1,\dots,N} \mathcal{J}_{ij}(\mathbf{h}^{(k-1)}, \mathbf{w}, t), \tag{2.7}$$

$$\mathcal{J}_{ij}(\mathbf{h}^{(k-1)}, \mathbf{w}, t) = \begin{cases} \mathcal{J}_{ij}^{(in)}, \text{ if } t \in [(\mathbf{x}_i, \mathbf{w}), (\mathbf{x}_j, \mathbf{w})]; \\ \mathcal{J}_{ij}^{(out)}, \text{ otherwise;} \end{cases}$$

and the procedure for optimal threshold search can be implemented more efficiently. For this case (2.7) we propose the special algorithm (Algorithm 3, Fig. 2.6), which requires $O(N^2)$ computations, and the number of computations for each pair from the training set is low enough. For the fixed hypothesis $\mathbf{w} = \mathbf{w}_k$, we arrange

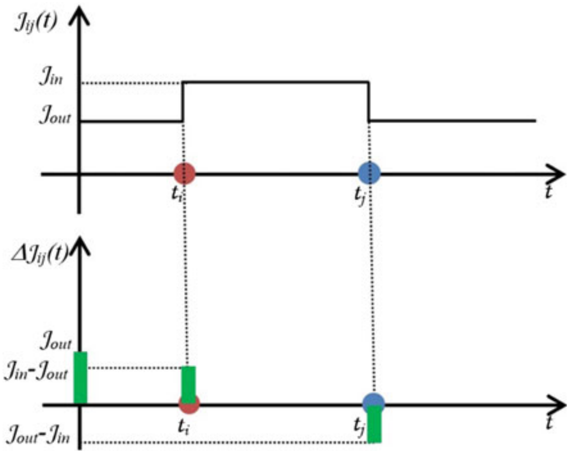


Fig. 2.6 Optimal threshold selection (Algorithm 3): stepwise objective function recovering via accumulation of step values from *left* to *right*

projections $t^{(k)}{}_i = (\mathbf{x}_i, \mathbf{w}_k)$ by increasing and test them as possible threshold values via calculating the $\mathcal{J}(X, \mathbf{h}^{(k-1)}, \mathbf{w}_k, t^{(k)}{}_i)$. The idea of this algorithm is to calculate the step values at each projection point and then recover the stepwise objective function via accumulation of step values from left to right.

2.4.4 BHF: Boosted Hashing Forest

Our Learn1BitHash procedure (see Algorithm 1) contains the recursive call of Learn1ProjectionHash procedure (Algorithm 2). Consider the tessellation of X by n-bit coder: $\mathbf{X}_B = \{X_\mathbf{b}, \mathbf{b}\in\{0,1\}^n\}$, $X_\mathbf{b} = \{\mathbf{x}\in X\colon \mathbf{h}(\mathbf{x}) = \mathbf{b}\}$, $X = \cup_{\mathbf{b}\in\{0,1\}^n} X_\mathbf{b}$. The process of recursive coding is a dichotomy splitting of training set with finding the optimized elementary coder for each subset at each level of tessellation. So, the recursive coder for kth bit

$$h^{(k)}(\mathbf{x}, \mathbf{h}^{(k-1)}) = h(\mathbf{w}(\mathbf{h}^{(k-1)}(\mathbf{x})), t(\mathbf{h}^{(k-1)}(\mathbf{x})), \mathbf{x}), \tag{2.8}$$

```
Algorithm 4: Boosted Hashing Forest
Input data: $X$, $\mathcal{J}$, $n_{ORC}$, $n_{BHF}$.
Output data: $\mathbf{h}(\mathbf{x})$: $\mathbf{x}\in R^m \to \mathbf{y}\in\{0,1\}^n$.
Initialization:
  $l$:=0; $\mathbf{h}^{[1,0]}$:= ( ).
Repeat iterations:
  $l$:= $l$+1;
  Form the objective as a function of $l$-th coding tree:
    $\mathcal{J}^{[l]}(X, \mathbf{h}^{[l,l]}) = \mathcal{J}(X, \mathbf{h}^{[1,l-1]}, \mathbf{h}^{[l,l]})$;
  Learn $l$-th coding tree:
    $\mathbf{h}^{[l,l]}$ := GreedyORC($\mathcal{J}^{[l]}$, $X$, $n_{ORC}$);
  Add $l$-th coding tree to the hashing forest:
    $\mathbf{h}^{[1,l]}(\mathbf{x})$ := ($\mathbf{h}^{[1,l-1]}(\mathbf{x})$, $\mathbf{h}^{[l,l]}(\mathbf{x})$);
while $l<n_{ORC}$. // stop if the given size of coder is got
```

is a combination of $2^{(k-1)}$ thresholded projections

$$\begin{aligned} h^{(k)}(\mathbf{x}, \mathbf{h}^{(k-1)}) &= \text{Learn1BitHash}(\mathcal{J},\quad X, \mathbf{h}^{(k-1)}) \\ &= \{\text{Learn1ProjectionHash}(\mathcal{J},\quad X(\mathbf{h}^{(k-1)}, \mathbf{b}), \mathbf{h}^{(k-1)}), \mathbf{b} \in \{0, 1\}^{(k-1)}\}. \end{aligned} \tag{2.9}$$

Such recursive n-bit coder $\mathbf{h}(\mathbf{x})$ is a *tree* of thresholded projections (Fig. 2.7), which has much more recognition power relative to the n-bit sequence of thresholded projections.

We know that one coding tree cannot provide the fine recognition rate. Besides, the number of projections in a tree grows exponentially with tree depth. So, the training set of some fixed size allows learning the trees with some limited depth only. Due to this, we form the hashing forest via the boosting of hashing trees with optimization

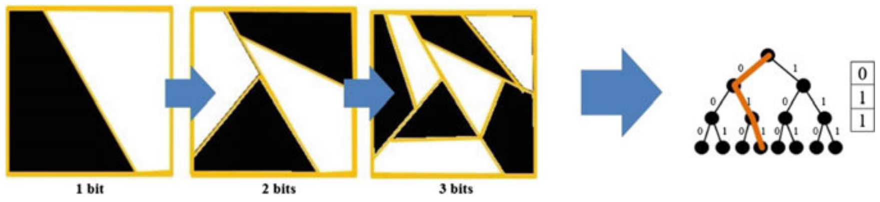


Fig. 2.7 The scheme of recursive coding by binary trees

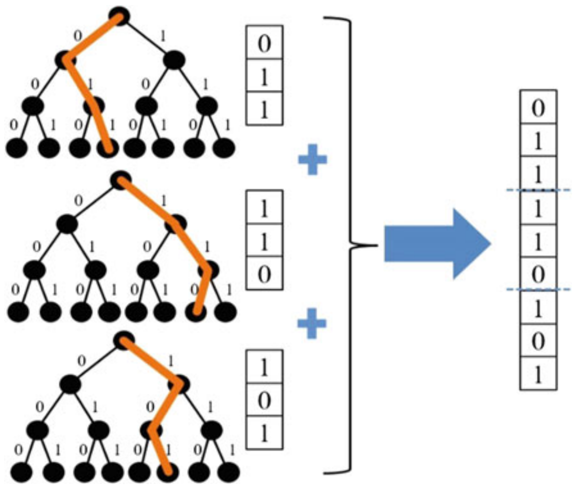


Fig. 2.8 Output binary code forming via concatenation of binary codes formed by trees of hashing forest (Algorithm 4)

of joint objective function for all trees. We call such approach as Boosted Hashing Forest (BHF) (Algorithm 4, Fig. 2.8).

Here we use the following notation: $n_{ORC} = p$ is a depth of coding tree; $n_{BHF} = n/p$ is a number of trees; $\mathbf{h}^{[1,l]} = (h^{(1)}(\mathbf{x}),\ldots,h^{(lp)}(\mathbf{x}))$, $\mathbf{h}^{[1,l-1]} = (h^{(1)}(\mathbf{x}),\ldots, h^{(lp-p)}(\mathbf{x}))$, $\mathbf{h}^{[l,l]} = (h^{(lp-p+1)}(\mathbf{x}),\ldots,h^{(lp)}(\mathbf{x}))$.

2.4.5 BHF: Hashing Forest as a Metric Space

We call the metric space (Y, d_Y) with d_Y: $Y \times Y \to R^+$ as *n-bit binary coded*, if the each $y \in Y$ corresponds to unique $\mathbf{b} \in \{0,1\}^n$, and two decoding functions are given: *feature decoder* $f_y(\mathbf{b})$: $\{0,1\}^n \to Y$ and *distance decoder* $f_d(\mathbf{b}_1,\mathbf{b}_2)$: $\{0,1\}^n \times \{0,1\}^n \to R^+$, $f_d(\mathbf{b}_1,\mathbf{b}_2) = d_Y(f_y(\mathbf{b}_1),f_y(\mathbf{b}_2))$. This allows define the *distance-based objective function* (DBOF) for coder $\mathbf{h}(\mathbf{x})$ of the form

$$\begin{gathered}(X, \mathbf{h}) \to min(\mathbf{h}) \Leftrightarrow \mathcal{J}(D_Y) \to min(D_Y),\\ D_Y = \{d_{ij} = f_d(\mathbf{h}(\mathbf{x}_i), \mathbf{h}(\mathbf{x}_j)), \mathbf{x}_i, \mathbf{x}_j \in X, \mathbf{h}(\mathbf{x}) \in \mathbf{H}\}_{i,j=1,\ldots,N}.\end{gathered} \tag{2.10}$$

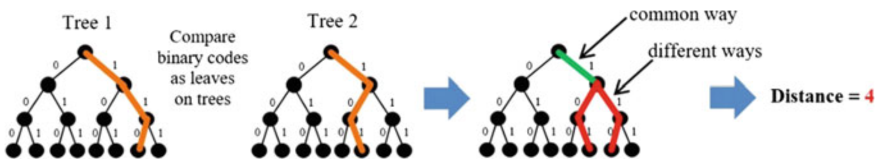


Fig. 2.9 Search index distance as a geodesic distance between codes as corresponding leaves on a coding tree

Such objective function depends on the set of coded distances d_{ij} only. In our current implementation of BHF we match p-bit binary trees via the *search index distance* (Fig. 2.9). It is a geodesic distance between codes as corresponding leaves on a coding tree

$$d_T(y_1, y_2) = f_{dT}(\mathbf{b}_1, \mathbf{b}_2) = 2\sum\nolimits_{k=1,\ldots,p}(1 - \prod\nolimits_{l=1,\ldots,k}(1 - |b_1^{(l)} - b_2^{(l)}|)). \tag{2.11}$$

Finally, we form a matching distance for total n-dimensional forest containing $q = n/p$ trees as a sum of distances between individual p-bit trees

$$d_{ij} = \sum\nolimits_{l=1,\ldots,q} f_{dT}(\mathbf{h}^{[l,l]}(\mathbf{x}_i), \mathbf{h}^{[l,l]}(\mathbf{x}_j)). \tag{2.12}$$

2.4.6 BHF: Objective Function for Face Verification and Identification

Let the similarity function s describes positive (authentic) and negative (imposter) pairs

$$s_{ij} = \begin{cases} 1, & \text{if class}(\mathbf{x}_i) = \text{class}\left(\mathbf{x}_j\right), \\ 0, & \text{otherwise.} \end{cases} \tag{2.13}$$

The "ideal" distance for k-bit binary code, is

$$g^{(k)}{}_{ij} = \begin{cases} 0, & \text{if } s_{ij} = 1, \\ d_{max}(k), & \text{otherwise,} \end{cases} \tag{2.14}$$

where $d_{max}(k)$ is a maximal possible distance. So, the *distance supervision objective function* can be formed as

$$\mathcal{J}_{Dist}(D_Y) = \sum\nolimits_{i=1,\ldots,N}\sum\nolimits_{j=1,\ldots,N} v_{ij}(d_{ij} - g_{ij})^2 \to min(D_Y = \{d_{ij}\}_{i,j=1,\ldots,N}), \tag{2.15}$$

where v_{ij} are the different weights for authentic and imposter pairs. This objective function (2.15) controls the verification performance (FAR and FRR).

In the identification-targeted biometric applications we need to control both distances and ordering of distances. Let $d^1{}_k = \max_l\{d_{kl}: s_{kl} = 1\}$ is a distance to the most far authentic and $d^0{}_k = \min_l\{d_{kl}: s_{kl} = 0\}$ is a distance to the closest imposter for the query $\mathbf{h}(\mathbf{x}_k)$. Then the ordering error e_{ij} for a pair $(\mathbf{x}_i,\mathbf{x}_j)$ can be expressed as

$$e_{ij} = \begin{cases} 1, \text{ if } (s_{ij} = 0\, \textit{and}\, h_{ij} < \max(d_i^1, d_j^1)) \\ \quad \text{or } (s_{ij} = 1\, \textit{and}\, h_{ij} > \min(d_i^0, d_j^0)) \\ \qquad 0, \text{ otherwise} \end{cases} \tag{2.16}$$

The ordering error occurs if imposter is closer than authentic or authentic is more far than imposter. So, the *distance order supervision objective function* can be formed as

$$\mathcal{J}_{Ord}(D_Y) = \sum\nolimits_{i=1,\dots,N} \sum\nolimits_{j=1,\dots,N} v_{ij}(d_{ij} - g_{ij})^2 e_{ij} \rightarrow min(D_Y = \{d_{ij}\}_{i,j=1,\dots,N}). \tag{2.17}$$

Here we penalize the difference between d_{ij} and objective distance g_{ij} like in (2.15), but only in case that the ordering error (2.16) occurs for this pair. So, criterion (2.17) directly controls the face identification characteristics (CMC).

Finally, for obtaining both verification and identification we combine the (2.15) and (2.17) resulting in

$$\begin{aligned} (D_Y) &= \alpha\, \mathcal{J}_{Dist}(D_Y) + (1-\alpha)\mathcal{J}_{Ord}(D_Y) \\ &= \sum\nolimits_{i=1,\dots,N} \sum\nolimits_{j=1,\dots,N} v_{ij}(d_{ij} - g_{ij})^2(e_{ij} + \alpha(1 - e_{ij})) \\ &\rightarrow min(D_Y = \{d_{ij}\}_{i,j=1,\dots,N}), \end{aligned} \tag{2.18}$$

where $\alpha \in [0,1]$ is a tuning parameter.

2.4.7 BHF Implementation for Learning Face Representation

For enhancement of our face representation learning we use some additional semi-heuristic modifications of described scheme. The goal distance (2.14) is modified

$$g^{(k)}{}_{ij} = \begin{cases} 0, \text{ if } s_{ij} = 1, \\ m^{(k-1)}{}_1 + 3\sigma^{(k-1)}{}_1, \text{ otherwise,} \end{cases} \tag{2.19}$$

where $m^{(k-1)}{}_1$ and $\sigma^{(k-1)}{}_1$ are the mean value and standard deviation of authentic coded distances. Such goal distance (2.19) excludes the penalizing of imposter pairs, which could not be treated as authentic. In (2.18) we use the adaptive weighting of pairs at each kth step of boosting

$$v^{(k)}{}_{ij} = \begin{cases} \gamma / a^{(k)}, \text{ if } s_{ij} = 1, \\ 1/b^{(k)}, \text{ otherwise}, \end{cases} \tag{2.20}$$

$$\begin{aligned} a^{(k)} &= \textstyle\sum_{i=1,\dots,N} \sum_{j=1,\dots,N} s_{ij}(d_{ij} - g_{ij})^2(e_{ij} + \alpha(1 - e_{ij})), \\ b^{(k)} &= \textstyle\sum_{i=1,\dots,N} \sum_{j=1,\dots,N} (1 - s_{ij})(d_{ij} - g_{ij})^2(e_{ij} + \alpha(1 - e_{ij})), \end{aligned} \tag{2.21}$$

where $a^{(k)}$ and $b^{(k)}$ provide the basic equal weight for all authentic and imposter pairs, and tuning parameter $\gamma > 1$ gives the slightly larger weights to authentic pairs.

We split the input m-dimensional feature vector to the set of independently coded subvectors with fixed sizes from the set $\mathbf{m} = \{m_{\min},\dots,m_{\max}\}$. At the each step of boosting we get the subvector with corresponding BHF elementary coder providing the best contribution to the objective function. The output binary vector of size n consists of some independently grown parts of size $n_{BHF} < n$. Such learning strategy prevents the premature saturation of objective function.

So, our binary face hashing is implemented with the following set of free parameters: $\mathbf{m}$, n_{ORC}, n_{BHF}, k_{RANSAC}, α and γ. The type of coded metrics is a free parameter of our approach too.

2.5 Experiments

In this section, we describe our methodology for learning and testing CNHF, report our results in Hamming embedding task, compare proposed BHF to original Boosted SSC, explore the CNHF performance w.r.t. depth of coding trees and compare CNHL and CNHF to best methods on LFW. We test the verification accuracy by the standard LFW unrestricted with outside labeled data protocol. Our CMC and rank-1 tests follow the methodology described in [2].

2.5.1 Methodology: Learning and Testing CNHF

The basic CNN is trained on CASIA-WebFace dataset. Face images are aligned by rotation of eye points to horizontal position with fixed eye-to-eye distance and crop to 128 × 128 size. The open source deep learning framework Caffe (http://caffe.berkeleyvision.org/) is used for training the basic CNN for multi-class face identification in the manner [24, 31]. The hashing forest is trained on the dataset containing 1,000 authentic pairs and correspondingly 999,000 imposter pairs of Faces in the Wild images (not from the testing LFW set). Finally, the family of CNHF coders is formed by proposed BHF: Hamming embedding coders 2,000 × 1 bit (250 byte), 200 × 1 bit (25 byte) and 32 × 1 bit (4 byte) of size; Hashing forest coders containing

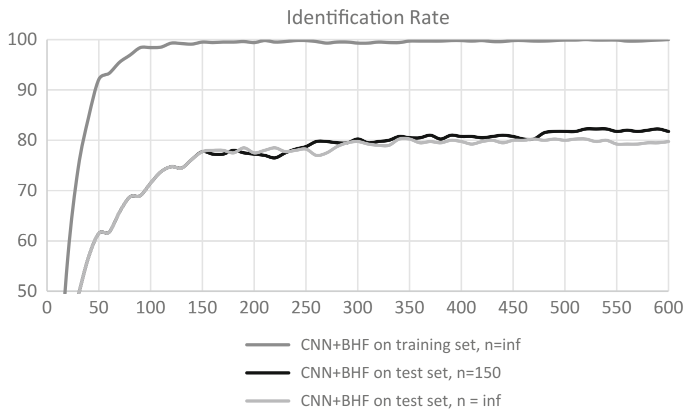


Fig. 2.10 Example of n_{BHF} parameter selection

2,000 trees with 2–7 bits depth (0.5–1.75 Kbyte of size). We used the common setting of BHF parameters: $\mathbf{m} = \{8, 16, 32\}$, $k_{RANSAC} = 50$, $\alpha = 0.25$, $\gamma = 1.1$. But we set $n_{BHF} = 200$ for CNN+BHF-200 × 1, $n_{BHF} = 500$ for CNN+BHF-2,000 × 1 and $n_{BHF} = 100$ for CNHF-2,000 × 7. Such n_{BHF} parameter values are determined experimentally based on the analysis of the speed of identification rate growing w.r.t. number of code bits in the hashing process. We determine the minimal number of generated code bits, which provides the best identification rate on training database in the hashing process. Figure 2.10 demonstrates the example of n_{BHF} parameter selection. Graphs for identification score w.r.t. number of coding trees are shown both for training and for testing set. One can see that on the training set the identification stabilizes approximately at the level of 150 coding trees. Correspondingly in testing the identification rate for $n_{BHF} = 150$ (600 coding trees are divided into four independent coding forests) outperforms the identification rate for $n_{BHF} = \infty$ (600 coding trees are not divided to independent parts) by ≈2%. The evaluation is performed on the Labeled Faces in the Wild (LFW) dataset. All the images in LFW dataset are processed by the same pipeline as in [11] and normalized to 128 × 128.

2.5.2 *Hamming Embedding: CNHL Versus CNN, BHF Versus Boosted SSC*

In this subsection, we test our approach in Hamming embedding task, so, CNHF degrades to CNHL. We compare CNHL to basic CNN on LFW via verification accuracy and ROC curve (Table 2.4 and Fig. 2.11a). The CNN face representation is formed like in [34] as a vector of activations of 256 top hidden layer neurons. The cosine similarity (CNN+CS) and L2-distance (CNN+L2) are applied for matching. CNHL coders 2,000 and 200 bit of size are trained by BHF and matched by

Table 2.4 Verification accuracy on LFW, code size, and matching speed of CNN and CNHL

Solution	Accuracy	Template size	Matches in sec
CNN+L2	0.947	8,192 bit	2,713,222
CNN+BHF-200×1	0.963	**200 bit**	**194,986,071**
CNN+CS	0.975	8,192 bit	2,787,632
CNN+BHF-2000×1	**0.9814**	**2,000 bit**	**27,855,153**

Hamming distance (CNN+BHF-2,000 × 1 and CNN+BHF-200 × 1 correspondingly). Our solution CNN+BHF-2,000 × 1 achieves verification accuracy 98.14% on LFW, which outperforms all other CNN-based solutions. Moreover, our 25-byte length solution CNN+BHF-200 × 1 outperforms CNN+L2. Table 2.4 additionally demonstrates the gain in template size and matching speed.

We compare CNHL trained by BHF to CNHL trained by original Boosted SSC. Figure 2.11c demonstrates that proposed BHF essentially outperforms Boosted SSC in identification (rank-1) on LFW for all binary template sizes. The maximal rank-1 is 0.91 for BHF-2,000 × 1 and 0.865 for BoostSSC-2,000 × 1 (relative to 0.899 for CNN+CS). The ROC graph for CNN+BHF is monotonously better than for CNN+BoostSSC with same template size (Fig. 2.11a). Figure 2.11b contains the CMC graphs (ranks 1–10), which demonstrate that BHF outperforms Boosted SSC with same template size (additionally note that CNN+BHF-2,000 × 1 outperforms CNN+CS).

2.5.3 *CNHF: Performance w.r.t. Depth of Trees*

CNHF with 2,000 output features formed by 7-bit coding trees (CNHF-2,000 × 7) achieves 98.59% on LFW. The identification result of CNHF-2,000 × 7 is 93% rank-1 on LFW relative to 89.9% rank-1 for CNN+CS. Figure 2.11f presents the ROC curves for CNHF with different depth coding trees. The forest with 7-bit coding trees is the best by ROC, but 6-bit and 5-bit depth solutions are very close. We suppose that the reason of this result is a limited amount of hashing forest training set. Figure 2.11d, e demonstrates that CNHF-2,000 × 7 outperforms basic CNN+CS and CNHF-2,000 × 1 both in verification (ROC) and in identification (CMC). So, we can conclude that the adding of hashing forest on the top of CNN allows both generating the compact binary face representation and increasing the face verification and especially identification rates.

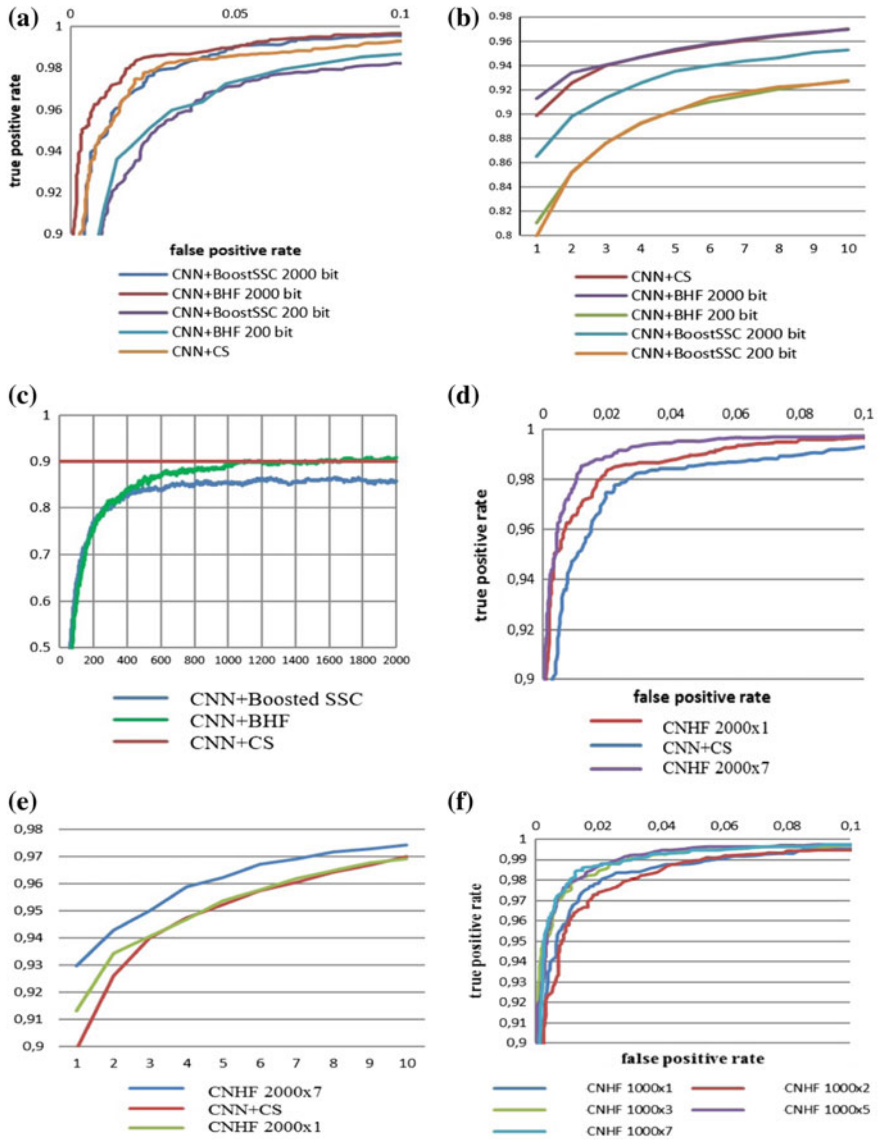


Fig. 2.11 **a** ROC curves, **b** CMC curves, **c** identification performance (rank 1) on LFW relative to the size of biometric template in bits for proposed BHF(CNN+BHF) and original Boosted SSC(CNN +BoostSSC) and best basic CNN solution without hashing - CNN + Last hidden layer + cosine similarity (CNN+CS), **d** ROC curves, **e** CMC curves for CNN+CS, CNHF-2000 × 1, CNHF-2000 × 7, **f** ROC curves for CNHF-1000 × p-bit trees

Table 2.5 Verification accuracy on LFW

Method	Accuracy
WebFace [24]	0.9613
CNHL-200×1	**0.963 ± 0.00494**
DeepFace-ensemble [21]	0.9730 ± 0.0025
DeepID [19]	0.9745 ± 0.0026
MFM Net [25]	0.9777
CNHL-2000×1	**0.9814**
CNHF-2000×7	**0.9859**
DeepID2 [17]	0.9915 ± 0.0013
DeepID3 [18]	0.9953 ± 0.0010
Baidu [11]	0.9977 ± 0.0006

2.5.4 CNHL and CNHF Versus Best Methods on LFW

We compare our CNHF solutions to state-of-the-art methods (best on LFW) via verification accuracy (Table 2.5). CNHF-2,000 × 1 outperforms DeepFace-ensemble [30], DeepID [27], WebFace [35] and MFM Net [34]. The DeepID2 [24], DeepID3 [26] and Baidu [14] multi-patch CNNs outperform our CNHF-2,000 × 1 based on single net.

Note that our CNHF-200 × 1 (25 byte) hash demonstrates 96.3% on LFW. Compare this result to previous best CNHL result [6]. On the one hand, the extreme-short 32-bit binary face representation [6] achieves 91% verification on LFW. Our CNHF 32 × 1 provides 90% only. On the other hand, face representation [6] requires 1000 bit for achieving the 96% verification on LFW. So, our CNHF-200 × 1 solution improves this face packing result in 5 times.

The identification result (rank-1) of our real-time coder CNHF-2,000 × 7 is 0.93 on LFW. It is close enough to best reported identification result of essentially deeper and slower multi-patch DeepID3 CNN [25] (0.96 rank-1 on LFW). Baidu [13] declares even better result (0.98 rank-1 on LFW), but they use the training set 1.2 million images of size w.r.t. 400 thousand images in our case.

2.6 Conclusion and Discussion

We develop the family of CNN-based binary face representations for real-time face identification. Our Convolutional Network with Hashing Forest (CNHF) generates binary face templates at the rate of 40+ fps with CPU Core i7 and 120+ fps with GPU GeForce GTX 650. Our 2,000 × 1-bit face coder provides the compact face coding (250 byte) with simultaneous increasing of verification (98.14%) and identification (91% rank-1) on LFW. Our 200 × 1-bit face coder provides the 40-time gain in template size and 70-time gain in a matching speed with 1% decreasing of verification accuracy relative to basic CNN (96.3% on LFW). Our CNHF with 2000 output 7-bit coding trees (CNHF-2,000 $\times$ 7) achieves 98.59% verification accuracy and 93% rank-1 on LFW (add 3% to rank-1 of basic CNN).

We use the multiple convolution deep network architecture for acceleration of source Max-Feature-Map (MFM) CNN architecture [31]. We propose and implement the new binary hashing technique, which forms the output feature space with given metric properties via joint optimization of face verification and identification. This Boosted Hashing Forest (BHF) technique combines the algorithmic structure of Boosted SSC approach and the binary code structure of forest hashing. Our experiments demonstrate that BHF essentially outperforms the original Boosted SSC in face identification test.

In the future we will try to achieve the better recognition rates via CNHF based on multi-patch CNN, which we can use for nonreal-time applications. We will evolve and apply the proposed BHF technique for different data coding and dimension reduction problems (supervised, semi-supervised and unsupervised). Additionally, we will investigate the influence of the output metric space properties in the process of hashing forest learning.

Acknowledgements This work is supported by grant from Russian Science Foundation (Project No. 16-11-00082).

References

1. M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in *Proceedings of the NIPS*, vol. 14 (2001), pp. 585–591
2. L. Best-Rowden, H. Han, C. Otto, B. Klare, A.K. Jain, Unconstrained face recognition: identifying a person of interest from a media collection. IEEE Trans. Inf. Forensic Secur. **9**(12), 2144–2157 (2014)
3. Z. Cao, Q. Yin, X. Tang, J. Sun, Face recognition with learning-based descriptor, in *Proceedings of the CVPR* (2010), pp. 2707–2714
4. D. Chen, X. Cao, F. Wen, J. Sun, Blessing of dimensionality: high-dimensional feature and its efficient compression for face verification, in *Proceedings of the CVPR* (2013), pp. 3025–3032
5. H. Fan, Z. Cao, Y. Jiang, Q. Yin, C. Doudou, Learning deep face representation (2014), arXiv:1403.2802
6. H. Fan, M. Yang, Z. Cao, Y. Jiang, Q. Yin, Learning compact face representation: packing a face into an int32, in *Proceedings of the ACM International Conference on Multimedia* (2014), pp. 933–936
7. A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in *Proceedings of the VLDB* (1999), pp. 518–529
8. Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval. IEEE Trans. Pattern Anal. Mach. Intell. **35**(12), 2916–2929 (2012)
9. K. Grauman, R. Fergus, Learning binary hash codes for large-scale image search, *Machine Learning for Computer Vision* (Springer, Berlin, 2013), pp. 49–87
10. K. He, F. Wen, J. Sun, K-means hashing: an affinity-preserving quantization method for learning binary compact codes, in *Proceedings of the CVPR* (2013), pp. 2938–2945

11. G.-B. Huang, M. Mattar, H. Lee, E. Learned-Miller, Learning to align from scratch, in *Proceedings of the NIPS*, vol. 25 (2012)
12. G. Irie, L. Zhenguo, W. Xiao-Ming, C. Shih-Fu, Locally linear hashing for extracting non-linear manifolds, in *Proceedings of the CVPR* (2014), pp. 2115–2122
13. J. Liu, Y. Deng, T. Bai, Z. Wei, C. Huang, Targeting ultimate accuracy: face recognition via deep embedding (2015), arXiv:1506.07310
14. W. Liu, J. Wang, R. Ji, Y.-G. Jiang, S.-F. Chang, Supervised hashing with kernels, in *Proceedings of the CVPR* (2012), pp. 2074–2081
15. Y. Mishina, M. Tsuchiya, H. Fujiyoshi, Boosted random forest. IEICE Trans. **E98D**(9), 1630–1636 (2015)
16. H.-V. Nguyen, L. Bai, Cosine similarity metric learning for face verification, in *Proceedings of the ACCV* (2010), pp. 709–720
17. Q. Qiu, G. Sapiro, A. Bronstein, Random forests can hash (2014), arXiv:1412.5083
18. R. Salakhutdinov, G. Hinton, Semantic hashing. Int. J. Approx. Reason. **50**(7), 969–978 (2009)
19. F. Schroff, D. Kalenichenko, J. Philbin, FaceNet: a unified embedding for face recognition and clustering, in *Proceedings of the CVPR* (2015), pp. 815–823
20. G. Shakhnarovich, Learning task-specific similarity, Ph.D. thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge (2005)
21. G. Shakhnarovich, P. Viola, T. Darrell, Fast pose estimation with parameter sensitive hashing. Proc. Comput. Vis. **2**, 750–757 (2003)
22. J. Springer, X. Xin, Z. Li, J. Watt, A. Katsaggelos, Forest hashing: expediting large scale image retrieval, in *Proceedings of the ICASSP* (2013), pp. 1681–1684
23. Y. Sun, X. Wang, X. Tang, Deep learning face representation by joint identification-verification, in *Proceedings of the NIPS*, vol. 27 (2014)
24. Y. Sun, X. Wang, X. Tang, Deep learning face representation from predicting 10,000 classes, in *Proceedings of the CVPR* (2014), pp. 1891–1898
25. Y. Sun, X. Wang, X. Tang, DeepID3: face recognition with very deep neural networks (2015), arXiv:1502.00873
26. Y. Taigman, L. Wolf, T. Hassner, Multiple one-shots for utilizing class label information, in *Proceedings of the BMVC* (2009)
27. Y. Taigman, M. Yang, M. Ranzato, L. Wolf, DeepFace: closing the gap to human-level performance in face verification, in *Proceedings of the CVPR* (2014), pp. 1701–1708
28. C. Vens, F. Costa, Random forest based feature induction, in *Proceedings of the ICDM* (2011), pp. 744–753
29. W. Wang, J. Yang, J. Xiao, S. Li, D. Zhou, Face recognition based on deep learning, in *Proceedings of the HCC*, vol. 8944 (2015), pp. 812–820
30. Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in *Proceedings of the NIPS*, vol. 21 (2008)
31. X. Wu, Learning robust deep face representation (2015), arXiv:1507.04844
32. D. Yi, Z. Lei, S. Liao, S.Z. Li, Learning face representation from scratch (2014), arXiv:1411.7923
33. G. Yu, J. Yuan, Scalable forest hashing for fast similarity search, in *Proceedings of the ICME* (2014), pp. 1–6
34. L. Zhang, Y. Zhang, X. Gu, J. Tang, Q. Tian, Topology preserving hashing for similarity search, in *Proceedings of the ACM International Conference on Multimedia* (2013), pp. 123–132
35. E. Zhou, Z. Cao, Q. Yin, Naive-deep face recognition: touching the limit of LFW benchmark or not? (2015), arXiv:1501.04690

Chapter 3
CMS-RCNN: Contextual Multi-Scale Region-Based CNN for Unconstrained Face Detection

Chenchen Zhu, Yutong Zheng, Khoa Luu and Marios Savvides

Abstract Robust face detection in the wild is one of the ultimate components to support various facial related problems, i.e., unconstrained face recognition, facial periocular recognition, facial landmarking and pose estimation, facial expression recognition, 3D facial model construction, etc. Although the face detection problem has been intensely studied for decades with various commercial applications, it still meets problems in some real-world scenarios due to numerous challenges, e.g., heavy facial occlusions, extremely low resolutions, strong illumination, exceptional pose variations, image or video compression artifacts, etc. In this paper, we present a face detection approach named Contextual Multi-Scale Region-based Convolution Neural Network (CMS-RCNN) to robustly solve the problems mentioned above. Similar to the region-based CNNs, our proposed network consists of the region proposal component and the region-of-interest (RoI) detection component. However, far apart of that network, there are two main contributions in our proposed network that play a significant role to achieve the state-of-the-art performance in face detection. First, the multi-scale information is grouped both in region proposal and RoI detection to deal with tiny face regions. Second, our proposed network allows explicit body contextual reasoning in the network inspired from the intuition of human vision system. The proposed approach is benchmarked on two recent challenging face detection databases, i.e., the WIDER FACE Dataset which contains high degree of variability, as well as the Face Detection Dataset and Benchmark (FDDB). The experimental results show that our proposed approach trained on WIDER FACE

C. Zhu (✉) · Y. Zheng (✉) · K. Luu · M. Savvides
CyLab Biometrics Center and the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: chenchez@andrew.cmu.edu

Y. Zheng
e-mail: yutongzh@andrew.cmu.edu

K. Luu
e-mail: kluu@andrew.cmu.edu

M. Savvides
e-mail: msavvid@ri.cmu.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*, Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_3

Dataset outperforms strong baselines on WIDER FACE Dataset by a large margin, and consistently achieves competitive results on FDDB against the recent state-of-the-art face detection methods.

3.1 Introduction

Detection and analysis on human subjects using facial feature based biometrics for access control, surveillance systems, and other security applications have gained popularity over the past few years. Several such biometrics systems are deployed in security checkpoints across the globe with more being deployed every day. Particularly, face recognition has been one of the most popular biometrics modalities attractive to security departments. Indeed, the uniqueness of facial features across individuals can be captured much more easily than other biometrics. In order to take into account a face recognition algorithm, however, face detection usually needs to be done first.

The problem of face detection has been intensely studied for decades with the aim of ensuring the generalization of robust algorithms to unseen face images [2–13]. Although the detection accuracy in recent face detection algorithms [14–19] has been highly improved due to the advancement of deep Convolutional Neural Networks (CNN), they are still far from achieving the same detection capabilities as a human due to a number of challenges in practice. For example, off-angle faces, large occlusions, low-resolutions and strong lighting conditions, as shown in Fig. 3.1, are always the important factors that need to be considered.

This paper presents an advanced CNN-based approach named Contextual Multi-Scale Region-based CNN (CMS-RCNN) to handle the problem of face detection in digital face images collected under numerous challenging conditions, e.g., heavy facial occlusion, illumination, extreme off-angle, low resolution, scale difference, etc. Our designed region-based CNN architecture allows the network to simultaneously look at multi-scale features, as well as to explicitly look outside facial regions as the potential body regions. In other words, this process tries to mimic the way of face detection by human in a sense that when humans are not sure about a face, seeing the body will increase our confidence. Additionally this architecture also helps to synchronize both the global semantic features in high-level layers and the localization features in low-level layers for facial representation. Therefore, it is able to robustly deal with the challenges in the problem of unconstrained face detection.

Our CMS-RCNN method introduces the Multi-Scale Region Proposal Network (MS-RPN) to generate a set of region candidates and the Contextual Multi-Scale Convolution Neural Network (CMS-CNN) to do inference on the region candidates of facial regions. A confidence score and bounding-box regression are computed for every candidate. In the end, the face detection system is able to decide the quality of the detection results by thresholding these generated confidence scores in given face images. The architecture of our proposed CMS-RCNN network for unconstrained face detection is illustrated in Fig. 3.2.

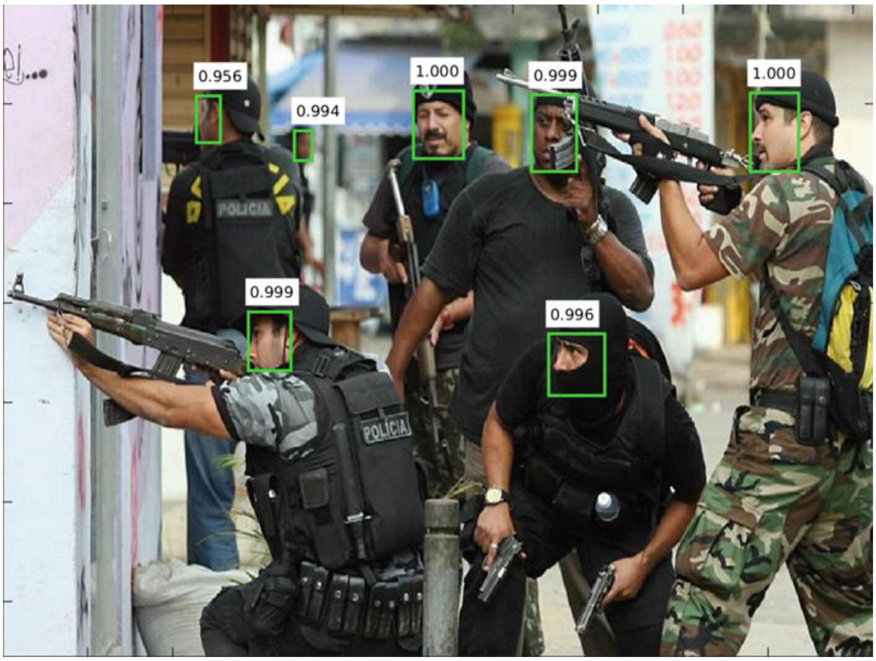


Fig. 3.1 An example of face detection results using our proposed CMS-RCNN method. The proposed method can robustly detect faces across occlusion, facial expression, pose, illumination, scale and low-resolution conditions from WIDER FACE Dataset [1]

Our approach is evaluated on two challenging face detection databases and compared against numerous recent face detection methods. First, the proposed CMS-RCNN method is compared against four strong baselines [1, 11, 16] on the WIDER FACE Dataset [1], a large-scale face detection benchmark database. This experiment shows its capability to detect face images in the wild, e.g., under occlusions, illumination, facial poses, low-resolution conditions, etc. Our method outperforms the baselines by a huge margin in all easy, medium, and hard partitions. It is also benchmarked on the Face Detection Data Set and Benchmark (FDDB) [20], a dataset of face regions designed for studying the problem of unconstrained face detection. The experimental results show that the proposed CMS-RCNN approach consistently achieves highly competitive results against the other state-of-the-art face detection methods.

The rest of this paper is organized as follows. In Sect. 3.2, we summarize prior work in face detection. Section 3.3 reviews a general deep learning framework, the background as well as the limitations of the Faster R-CNN in the problem of face detection. In Sect. 3.4, we introduce our proposed CMS-RCNN approach for the problem of unconstrained face detection. Section 3.5 presents the experimental face detection results and comparisons obtained using our proposed approach on two challenging face detection databases, i.e., the WIDER FACE Dataset and the FDDB database. Finally, our conclusions in this work are presented in Sect. 3.6.

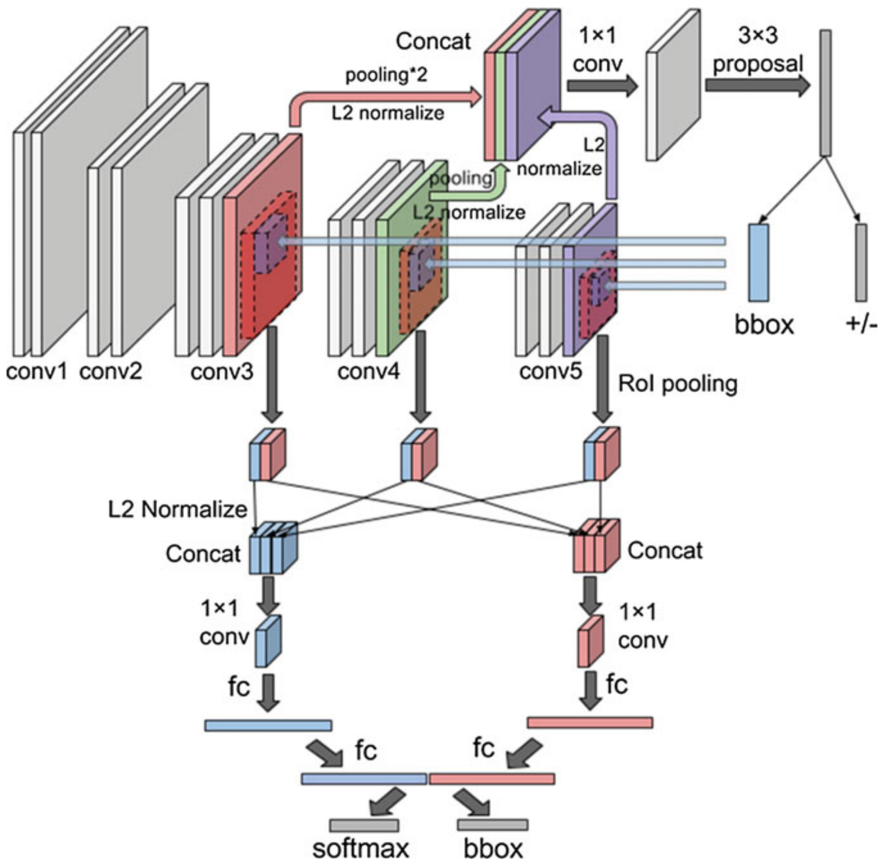


Fig. 3.2 Our proposed Contextual Multi-Scale Region-based CNN model. It is based on the VGG-16 model [21], with five sets of convolution layers in the middle. The upper part is the Multi-Scale Region Proposal Network (MS-RPN) and the lower part is the Contextual Multi-Scale Convolution Neural Network (CMS-CNN). In the CMS-CNN, the face features labeled as *blue blocks* and the body context features labeled as *red blocks* are processed in parallel and combined in the end for final outputs, i.e., confidence score and bounding box

3.2 Related Work

Face detection has been a well-studied area of computer vision. One of the first well-performing approaches to the problem was the Viola–Jones face detector [2]. It was capable of performing real-time face detection using a cascade of boosted simple Haar classifiers. The concepts of boosting and using simple features has been the basis for many different approaches [3] since the Viola–Jones face detector. These early detectors tended to work well on frontal face images but not very well on faces in different poses. As time has passed, many of these methods have been able to deal with off-angle face detection by utilizing multiple models for the various poses of the face. This increases the model size but does afford more practical uses of the methods. Some approaches have moved away from the idea of simple features but continued to use the boosted learning framework. Li and Zhang [5] used SURF cascades for general object detection but also showed good results on face detection.

More recent work on face detection has tended to focus on using different models such as a Deformable Parts Model (DPM) [4, 22]. Zhu and Ramanan's work was an interesting approach to the problem of face detection where they combined the problems of face detection, pose estimation, and facial landmarking into one framework. By utilizing all three aspects in one framework, they were able to outperform the state of the art at the time on real-world images. Yu et al. [23] extended this work by incorporating group sparsity in learning which landmarks are the most salient for face detection as well as incorporating 3D models of the landmarks in order to deal with pose. Chen et al. [10] have combined ideas from both of these approaches by utilizing a cascade detection framework while simultaneously localizing features on the face for alignment of the detectors. Similarly, Ghiasi and Fowlkes [12] have been able to use heirarchical DPMs not only to achieve good face detection in the presence of occlusion but also landmark localization. However, Mathias et al. [9] were able to show that both DPM models and rigid template detectors similar to the Viola–Jones detector have a lot of potential that has not been adequately explored. By retraining these models with appropriately controlled training data, they were able to create face detectors that perform similarly to other, more complex state-of-the-art face detectors.

All of these approaches to face detection were based on selecting a feature extractor beforehand. However, there has been work done in using a ConvNet to learn which features are used to detect faces. Neural Networks have been around for a long time but have been experiencing a resurgence in popularity due to hardware improvements and new techniques resulting in the capability to train these networks on large amounts of training data. Li et al. [14] utilized a cascade of CNNs to perform face detection. The cascading networks allowed them to process different scales of faces at different levels of the cascade while also allowing for false positives from previous networks to be removed at later layers in a similar approach to other cascade detectors. Yang et al. [16] approached the problem from a different perspective more similar to a DPM approach. In their method, the face is broken into several facial parts such as hair, eyes, nose, mouth, and beard. By training a detector on each part and combining the score maps intelligently, they were able to achieve accurate face detection even under occlusions. Both of these methods require training several networks in order to achieve their high accuracy. Our method, on the other hand, can be trained as a single network, end to end, allowing for less annotation of training data needed while maintaining highly accurate face detection.

The ideas of using contextual information in object detection have been studied in several recent work with very high detection accuracy. Divvala et al. [24] reviewed the role of context in a contemporary, challenging object detection in their empirical evaluation analysis. In their conclusions, the context information not only reduces the overall detection errors, but also the remaining errors made by the detector are more reasonable. Bell et al. [25] introduced an advanced object detector method named

Inside-Outside Network (ION) to exploit information both inside and outside the region of interest. In their approach, the contextual information outside the region of interest is incorporated using spatial recurrent neural networks. Inside the network, skip pooling is used to extract information at multiple scales and levels of abstraction. Recently, Zagoruyko et al. [26] have presented the MultiPath network with three modifications to the standard Fast R-CNN object detector, i.e., skip connections that give the detector access to features at multiple network layers, a foveal structure to exploit object context at multiple object resolutions, and an integral loss function and corresponding network adjustment that improve localization. The information in their proposed network can flow along multiple paths. Their MultiPath network is combined with DeepMask object proposals to solve the object detection problem.

Unlike all the previous approaches that select a feature extractor beforehand and incorporate a linear classifier with the depth descriptor beside RGB channels, our method solves the problem under a deep learning framework where the global and the local context features, i.e., multi scaling, are synchronized to Faster Region-based Convolutional Neural Networks in order to robustly achieve semantic detection.

3.3 Background in Deep Convolution Nets

The recent studies in deep ConvNets have achieved significant results in object detection, classification and modeling [27]. In this section, we review various well-known Deep ConvNets. Then, we show the current limitations of the Faster R-CNN, one of the state-of-the-art deep ConvNet methods in object detection, in the defined context of the face detection.

3.3.1 Region-Based Convolution Neural Networks

One of the most important approaches for the object detection task is the family of Region-based Convolution Neural Networks (R-CNN).

R-CNN [28], the first generation of this family, applies the high-capacity deep ConvNet to classify given bottom-up region proposals. Due to the lack of labeled training data, it adopts a strategy of supervised pretraining for an auxiliary task followed by domain-specific fine-tuning. Then the ConvNet is used as a feature extractor and the system is further trained for object detection with Support Vector Machines (SVM). Finally, it performs bounding-box regression. The method achieves high accuracy but is very time-consuming. The system takes a long time to generate region proposals, extract features from each image, and store these features in a hard disk, which also takes up a large amount of space. At testing time, the detection process takes 47 s per image using VGG-16 network [21] implemented in GPU due to the slowness of feature extraction. In other words, R-CNN is slow because it processes each object proposal independently without sharing computation.

Fast R-CNN [29] solves this problem by sharing the features between proposals. The network is designed to only compute a feature map once per image in a fully convolutional style, and to use ROI pooling to dynamically sample features from the feature map for each object proposal. The network also adopts a multitask loss, i.e., classification loss and bounding-box regression loss. Based on the two improvements, the framework is trained end to end. The processing time for each image significantly reduced to 0.3 s. Fast R-CNN accelerates the detection network using the ROI pooling layer. However the region proposal step is designed out of the network hence still remains a bottleneck, which results in suboptimal solution and dependence on the external region proposal methods.

Faster R-CNN [30] addresses the problem with fast R-CNN by introducing the Region Proposal Network (RPN). An RPN is implemented in a fully convolutional style to predict the object bounding boxes and the objectness scores. In addition, the anchors are defined with different scales and ratios to achieve the translation invariance. The RPN shares the full-image convolution features with the detection network. Therefore the whole system is able to complete both proposal generation and detection computation within 0.2 s using very deep VGG-16 model [21]. With a smaller ZF model [31], it can reach the level of real-time processing.

3.3.2 Limitations of Faster R-CNN

The Region-based CNN family, e.g., Faster R-CNN and its variants [29], achieves the state-of-the-art performance results in object detection on the PASCAL VOC dataset. These methods can detect objects such as vehicles, animals, people, chairs, etc., with very high accuracy. In general, the defined objects often occupy the majority of a given image. However, when these methods are tested on the challenging Microsoft COCO dataset [32], the performance drops a lot, since images contain more small, occluded, and incomplete objects. Similar situations happen in the problem of face detection. We focus on detecting only facial regions that are sometimes small, heavily occluded and of low resolution (as shown in Fig. 3.1).

The detection network in designed Faster R-CNN is unable to robustly detect such tiny faces. The intuition point is that the regions of interest pooling layer, i.e., ROI pooling layer, builds features only from the last single high-level feature map. For example, the global stride of the ‘conv5’ layer in the VGG-16 model is 16. Therefore, given a facial region with the sizes less than 16×16 pixels in an image, the projected ROI pooling region for that location will be less than 1 pixel in the ‘conv5’ layer, even if the proposed region is correct. Thus, the detector will have much difficulty to predict the object class and the bounding-box location based on information from only one pixel.

3.3.3 Other Face Detection Method Limitations

Other challenges in object detection in the wild include occlusion and low resolution. For face detection, it is very common for people to wear stuffs like sunglasses, scarf and hats, which occlude the face. In such cases, the methods that only extract features from faces do not work well. For example, Faceness [16] consider finding faces through scoring facial parts responses by their spatial structure and arrangement, which works well on clear faces. But when facial parts are missing due to occlusion or when face itself is too small, facial parts become more hard to detect. Therefore, the body context information plays its role. As an example of context-dependent objects, faces often come together with human body. Even though the faces are occluded, we can still locate it only by seeing the whole human body. Similar advantages for faces at low resolution, i.e., tiny faces. The deep features cannot tell much about tiny faces since their receptive field is too small to be informative. Introducing context information can extend the area to extract features and make them meaningful. On the other hand, the context information also helped with reducing false detection as discussed previously, since context information tells the difference between real faces with bodies and face-like patterns without bodies.

3.4 Contextual Multi-Scale R-CNN

Our goal is to detect human faces captured under various challenging conditions such as strong illumination, heavily occlusion, extreme off-angles, and low resolution. Under these conditions, the current CNN-based detection systems suffer from two major problems, i.e., (1) tiny faces are hard to identify; (2) only face region is taken into consideration for classification. In this section, we show why these problems hinder the ability of a face detection system. Then, our proposed network is presented to address these problems by using the Multi-Scale Region Proposal Network (MS-RPN) and the Contextual Multi-Scale Convolution Neural Network (CMS-CNN), as illustrated in Fig. 3.2. Similar to Faster R-CNN, the MS-RPN outputs several region candidates and the CMS-CNN computes the confidence score and bounding box for each candidate.

3.4.1 Identifying Tiny Faces

Why tiny faces are hard to be robustly detected by the previous region-based CNNs? The reason is that in these networks both the proposed region and the classification score are produced from one single high-level convolution feature map. This representation does not have enough information for the multiple tasks, i.e., region proposal and RoI detection. For example, Faster R-CNN generates region candidates and does RoI pooling from the ‘conv5’ layer of the VGG-16 model, which has a overall stride of 16. One issue is that the reception field in this layer is quite large. When the face size is less than 16-by-16 pixels, the corresponding output in ‘conv5’ layer is less than 1 pixel, which is insufficient to encode informative features. The other issue is that as the convolution layers go deeper, each pixel in the feature map gather more and more information outside the original input region so that it contains lower proportion of information for the region of interest. These two issues together make the last convolution layer less representative for tiny faces.

3.4.1.1 Multiple Scale Faster-RCNN

Our solution for this problem is a combination of both global and local features, i.e., multiple scales. In this architecture, the feature maps are incorporated from lower level convolution layers with the last convolution layer for both MS-RPN and CMS-CNN. Features from lower convolution layer help get more information for the tiny faces, because stride in lower convolution layer will not be too small. Another benefit is that both low-level feature with localization capability and high-level feature with semantic information are fused together [33], since face detection needs to localize the face as well as to identify the face. In the MS-RPN, the whole lower level feature maps are down-sampled to the size of high-level feature map and then concatenated with it to form a unified feature map. Then we reduce the dimension of the unified feature map and use it to generate region candidates. In the CMS-CNN, the region proposal is projected into feature maps from multiple convolution layers. And RoI pooling is performed in each layer, resulting in a fixed-size feature tensor. All feature tensors are normalized, concatenated and dimension-reduced to a single feature blob, which is forwarded to two fully connected layers to compute a representation of the region candidate.

3.4.1.2 L2 Normalization

In both MS-RPN and CMS-CNN, concatenation of feature maps is done with L2 normalization layer [34], shown in Fig. 3.2, since the feature maps from different layer have generally different properties in terms of numbers of channels, scale of value and norm of feature map pixels. Generally, comparing with values in shallower layers, the values in deeper layers are usually too small, which leads to the dominance of shallower layers. In practice, it is impossible for the system to readjust and tune value from each layer for best performance. Therefore, L2 normalization layers before concatenation are crucial for the robustness of the system because it keeps the value from each layer in roughly the same scale.

The normalization is performed within each pixel, and all feature map is treated independently

$$\hat{\mathbf{x}} = \frac{\mathbf{x}}{\|\mathbf{x}\|_2} \; \|\mathbf{x}\|_2 = \left(\sum_{i=1}^{d} |x_i|\right)^{\frac{1}{2}} \tag{3.1}$$

where the $\mathbf{x}$ and $\hat{\mathbf{x}}$ stand for the original pixel vector and the normalized pixel vector respectively. d stands for the number of channels in each feature map tensor.

During training, scaling factors γ_i will be updated to readjust the scale of the normalized features. For each channel i, the scaling factor follows:

$$y_i = \gamma_i \hat{x}_i \tag{3.2}$$

where y_i stand for the rescaled feature value.

Following the backpropagation and chain rule, the update for scaling factor γ is

$$\frac{\partial l}{\partial \hat{\mathbf{x}}} = \frac{\partial l}{\partial \mathbf{y}} \cdot \gamma \frac{\partial l}{\partial \mathbf{x}} = \frac{\partial l}{\partial \hat{\mathbf{x}}} \left(\frac{\mathbf{I}}{\|\mathbf{x}\|_2} - \frac{\mathbf{x}\mathbf{x}^T}{\|\mathbf{x}\|_2^3} \right) \frac{\partial l}{\partial \gamma_i} = \sum_{y_i} \frac{\partial l}{\partial y_i} \hat{x}_i \tag{3.3}$$

where $\mathbf{y} = [y_1, y_2, \ldots, y_d]^T$.

3.4.1.3 New Layer in Deep Learning Caffe Framework

The system integrate information from lower layer feature maps, i.e., third and fourth convolution layers, to extract determinant features for tiny faces. For both parts of our system, i.e., MS-RPN and CMS-CNN, the L2 normalization layers are inserted before concatenation of feature maps from the three layers. The features were rescaled to proper values and concatenated to a single feature map. We set the initial scaling factor in a special way, following two rules. First, the average scale for each feature map is roughly identical; second, after the following 1×1 convolution, the resulting tensor should have the same average scale as the conv5 layer in the work of Faster R-CNN. As implied, after the following 1×1 convolution, the tensor should be the same as the original architecture in Faster R-CNN, in terms of its size, scale of values, and function for the downstream process.

3.4.2 Integrating Body Context

When humans are searching for faces, they try to look for not only the facial patterns, e.g., eyes, nose, mouth, but also the human bodies. Sometimes a human body makes us more convinced about the existence of a face. In addition, sometimes human body

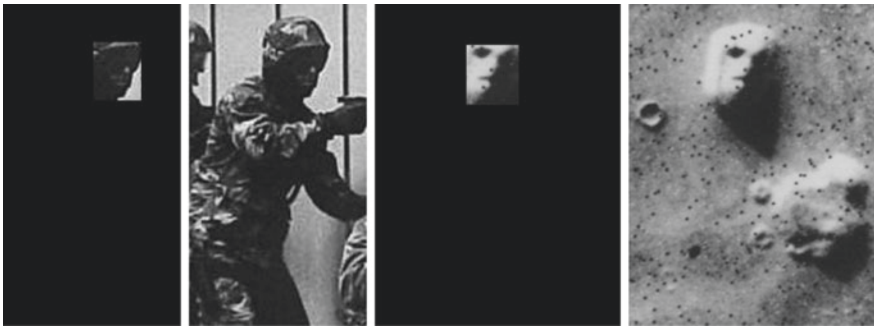


Fig. 3.3 Examples of body context helping face identification. The first two figures show that existence of a body can increase the confidence of finding a face. The last two figures show that what looks like a face turns out to be a mountain on the planet surface when we see more context information

helps to reject false positives. If we only look at face regions, we may make mistakes identifying them. For example, Fig. 3.3 shows two cases where body region plays a significant role for correct detection. This intuition is not only true for human but also valid in computer vision. Previous research has shown that contextual reasoning is a critical piece of the object recognition puzzle, and that context not only reduces the overall detection errors, but, more importantly, the remaining errors made by the detector are more reasonable [24]. Based on this intuition, our network is designed to make explicit reference to the human body context information in the RoI detection.

In our proposed network, the contextual body reasoning is implemented by explicitly grouping body information from convolution feature maps shown as the red blocks in Fig. 3.2. Specifically, additional RoI pooling operations are performed for each region proposal in convolution feature maps to represent the body context features. Then same as the face feature tensors, these body feature tensors are normalized, concatenated, and dimension-reduced to a single feature blob. After two fully connected layers the final body representation is concatenated with the face representation. They together contribute to the computation of confidence score and bounding-box regression.

With projected region proposal as the face region, the additional RoI pooling region represents the body region and satisfies a predefined spatial relation with the face region. In order to model this spatial relation, we make a simple hypothesis that if there is a face, there must exist a body, and the spatial relation between each face and body is fixed. This assumption may not be true all the time but should cover most of the scenarios since most people we see in the real world are either standing or sitting. Therefore, the spatial relation is roughly fixed between the face and the vertical body.

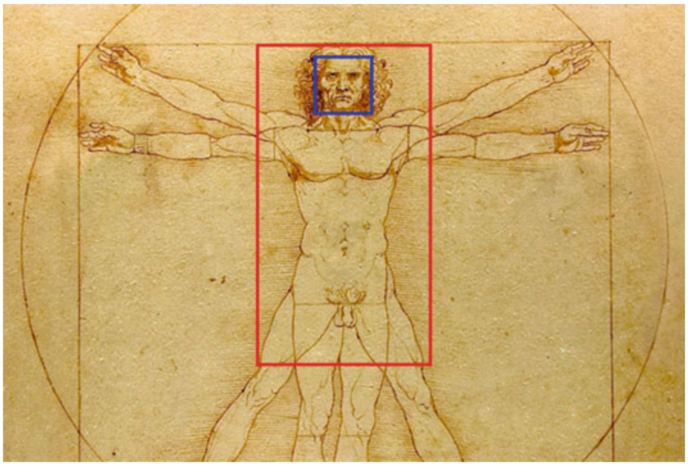


Fig. 3.4 The Vitruvian Man: spatial relation between the face (*blue box*) and the body (*red box*)

Mathematically, this spatial relation can be represented by four parameters presented in Eq. 3.4.

$$t_x = (x_b - x_f)/w_f t_y = (y_b - y_f)/h_f t_w = \log(w_b/w_f) t_h = \log(h_b/h_f) \quad (3.4)$$

where $x_{(*)}$, $y_{(*)}$, $w_{(*)}$, and $h_{(*)}$ denote the two coordinates of the box center, width, and height respectively. And b and f stand for body and face respectively. t_x, t_y, t_w, and t_h are the parameters. Through out this paper, we fix the for parameters such that the two projected RoI regions of face and body satisfies a certain spatial ratio illustrated in the famous drawing in Fig. 3.4.

3.4.3 Information Fusion

It's worth noticing that in our deep network architecture we have multiple face feature maps and body context feature maps for each proposed region. A critical issue is how we effectively fuse these information, i.e., what computation to apply and in which stage.

In our network, features extracted from different convolution layers need to be fused together to get a uniform representation. They cannot be naively concatenated due to the overall differences of the numbers of channels, scales of values, and norms of feature map pixels among these layers. The detailed research shows that the deeper layers often contain smaller values than the shallower layers. Therefore, the larger values will dominate the smaller ones, making the system rely too much on shallower features rather than a combination of multiple scale features causing the system to no longer be robust. We adopt the normalization layer from [34] to address this problem. The system takes the multiple scale features and apply L2 normalization along the channel axis of each feature map. Then, since the channel size is different among layers, the normalized feature map from each layer needed to be reweighted, so that their values are at the same scale. After that, the feature maps are concatenated to one single feature map tensor. This modification helps to stabilize the system and increase the accuracy. Finally, the channel size of the concatenated feature map is shrunk to fit right in the original architecture for the downstream fully connected layers.

Another crucial question is whether to fuse the face information and the body information at a early stage or at the very end of the network. Here we choose the late fusion strategy in which face features and body context features are extracted in two parallel pipelines. At the very end of the network two representations for face and body context are concatenated together to form a long feature vector. Then this feature vector is forwarded to compute confidence score and bounding-box regression. The other strategy is the early fusion, in which face feature maps and body context feature maps get concatenated right after RoI pooling and normalization. These two strategies combine the information from face and body context, but we prefer the late fusion. The reason is that we want the network to make decisions in a more semantic space. We care more about the existence of the face and the body. The localization information is already encoded in the predefined spatial relation mentioned in Sect. 3.4.2. Moreover empirical experiments also show that late fusion strategy works better.

3.4.4 Implementation Details

Our CMS-RCNN is implemented in the Caffe deep learning framework [35]. The first five sets of convolution layers have the same architecture as the deep VGG-16 model, and during training their parameters are initialized from the pretrained VGG-16. For simplicity we refer to the last convolution layers in set 3, 4 and 5 as 'conv3', 'conv4', and 'conv5' respectively. All the following layers are connected exclusively to these three layers. In the MS-RPN, we want 'conv3', 'conv4', and 'conv5' to be synchronized to the same size so that concatenation can be applied. So 'conv3' is followed by pooling layer to perform down-sampling. Then 'conv3', 'conv4', and 'conv5' are normalized along the channel axis to a learnable reweighting scale and concatenated together. To ensure training convergence, the initial reweighting scale needs to be carefully set. Here we set the initial scale of 'conv3', 'conv4', and 'conv5' to be 66.84, 94.52, and 94.52 respectively. In the CMS-CNN, the RoI pooling layer already ensure that the pooled feature maps have the same size. Again we normalize the pooled features to make sure the downstream values are at reasonable scales when training is initialized. Specifically, features pooled from 'conv3', 'conv4', and 'conv5' are initialized with scale to be 57.75, 81.67, and 81.67 respectively, for both face and body pipelines. The MS-RPN and the CMS-CNN share the same parameters for all convolution layers so that computation can be done once, resulting in higher efficiency. Additionally, in order to shrink the channel size of the concatenated feature map, a 1×1 convolution layer is then employed. Therefore the channel size of final feature map is at the same size as the original fifth convolution layer in Faster R-CNN.

3.5 Experiments

This section presents the face detection benchmarking using our proposed CMS-RCNN approach on the WIDER FACE dataset [1] and the Face Detection Data Set and Benchmark (FDDB) [20] database. The WIDER FACE dataset is experimented with high degree of variability. Using this database, our proposed approach robustly outperforms strong baseline methods, including Two-stage CNN [1], Multi-scale Cascade CNN [1], Faceness [16] and Aggregate Channel Features (ACF) [11], by a large margin. We also show that our model trained on WIDER FACE dataset generalizes well enough to the FDDB database. The trained model consistently achieves competitive results against the recent state-of-the-art face detection methods on this database, including HyperFace [19], DP2MFD [17], CCF [18], Faceness [16], NPDFace [13], MultiresHPM [12], DDFD [15], CascadeCNN [14], ACF-multiscale [11], Pico [7], HeadHunter [9], Joint Cascade [10], Boosted Exemplar [8], and PEP-Adapt [6].

3.5.1 Experiments on WIDER FACE Dataset

Data Description

WIDER FACE is a public face detection benchmark dataset. It contains 393,703 labeled human faces from 32,203 images collected based on 61 event classes from Internet. The database has many human faces with a high degree of pose variation, large occlusions, low resolutions, and strong lighting conditions. The images in this database are organized and split into three subsets, i.e., training, validation and testing. Each contains 40, 10, and 50% respectively of the original databases. The images and the ground truth labels of the training and the validation sets are available online for experiments. However, in the testing set, only the testing images (not the ground truth labels) are available online. All detection results are sent to the database server for evaluating and receiving the Precision-Recall curves.

In our experiments, the proposed CMS-RCNN is trained on the training set of the WIDER FACE dataset containing 159,424 annotated faces collected in 12,880 images. The trained model on this database are used in testing of all databases.

With Context Versus Without Context

As we show in Sect. 3.4.2 that human vision can benefit from additional context information for better detection and recognition, we show in this section how does explicit contextual reasoning in the network help improve the model performance.

To prove this, we test our models with and without body context information on the validation set of WIDER FACE dataset. The model without body context is implemented by removing the context pipeline and only use the representation from face pipeline to compute the confidence score and the bounding-box regression. We compare their performances as illustrated in Fig. 3.5. The Faster R-CNN method is setup as a baseline. We also compare our models with different strategies of information fusion discussed in Sect. 3.4.3, which we call early-fused context and late-fused context respectively.

Experiment results show that the model with late-fused context has significant improvement over the one without context, which means that they can handle the challenging conditions better. Moreover, we observe that early-fused context dose help detecting more faces because its curve is longer than the one without context. However, early-fused context gets fired on too many false positives which results in low AP score and a drop in the beginning of the curve. Figure 3.6 illustrates the false positives compared to late-fused context. This is probably because early-fused context concatenates all feature maps at a very early stage, resulting in a high-dimensional feature map. This can cause over-fitting due to the curse of dimension. Therefore, we fix our model to be late-fused context for other experiments.

To find out the minimum resolution at which our proposed network can detect a face, we set the threshold such that our detector can reach 90% precision in Fig. 3.5. Then we collect all the correct faces detected by our method in the WIDER FACE validation set. It turns out the minimum size of faces we can detect is around 10×10 pixels.

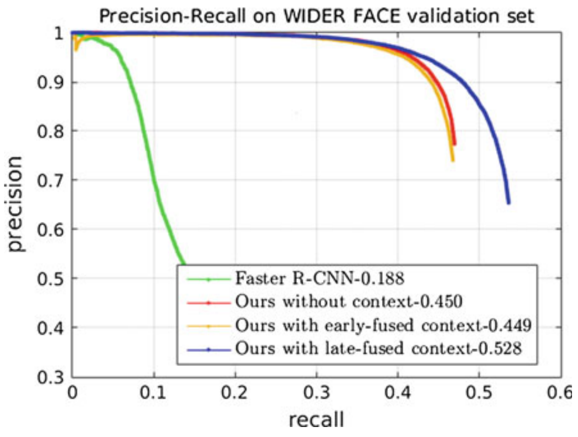


Fig. 3.5 Precision-Recall curves on the WIDER FACE validation set. The baseline (*green curve*) is generated by the Faster R-CNN [30] model trained on WIDER FACE training set. We show that our model without context (*red curve*) outperforms baseline by a large gap. With late-fused context, the performance gets boosted even further (*blue curve*). The numbers in the legend are the average precision values

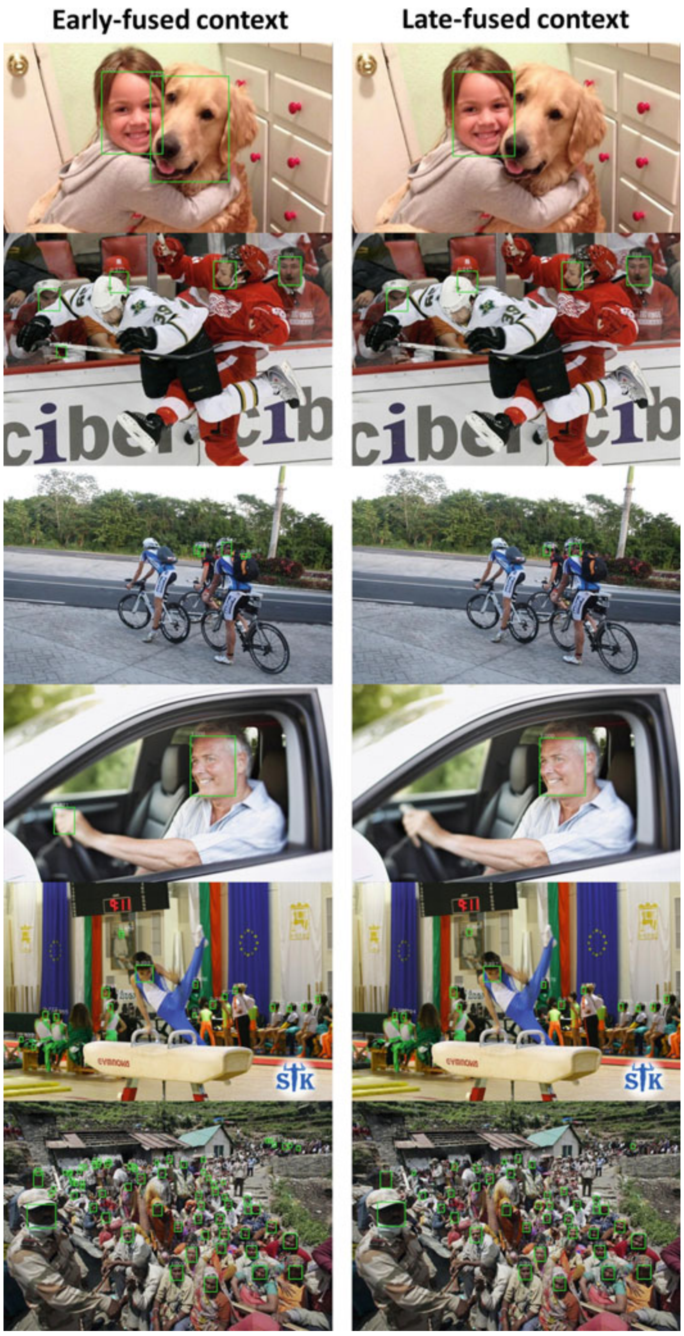


Fig. 3.6 Comparison between early-fused context and late-fused context on some examples in WIDER FACE validation set. It is clear that early-fused context is easier getting fired on some false positives, resulting in a lower AP score

Testing and Comparison

During the testing phase, the face images in the testing set are divided into three parts based on their detection rates on EdgeBox [36]. In other words, face images are divided into three levels according to the difficulties of the detection, i.e., Easy, Medium, and Hard [1]. The proposed CMS-RCNN model is compared against recent strong face detection methods, i.e., Two-stage CNN [1], Multi-scale Cascade CNN [1], Faceness [16], and Aggregate Channel Features (ACF) [11]. All these methods are trained on the same training set and tested on the same testing set.

The Precision-Recall curves and AP values are shown in Fig. 3.7. Our method outperforms those strong baselines by a large margin. It achieves the best average precision in all level faces, i.e., AP = 0.902 (Easy), 0.874 (Medium) and 0.643 (Hard),

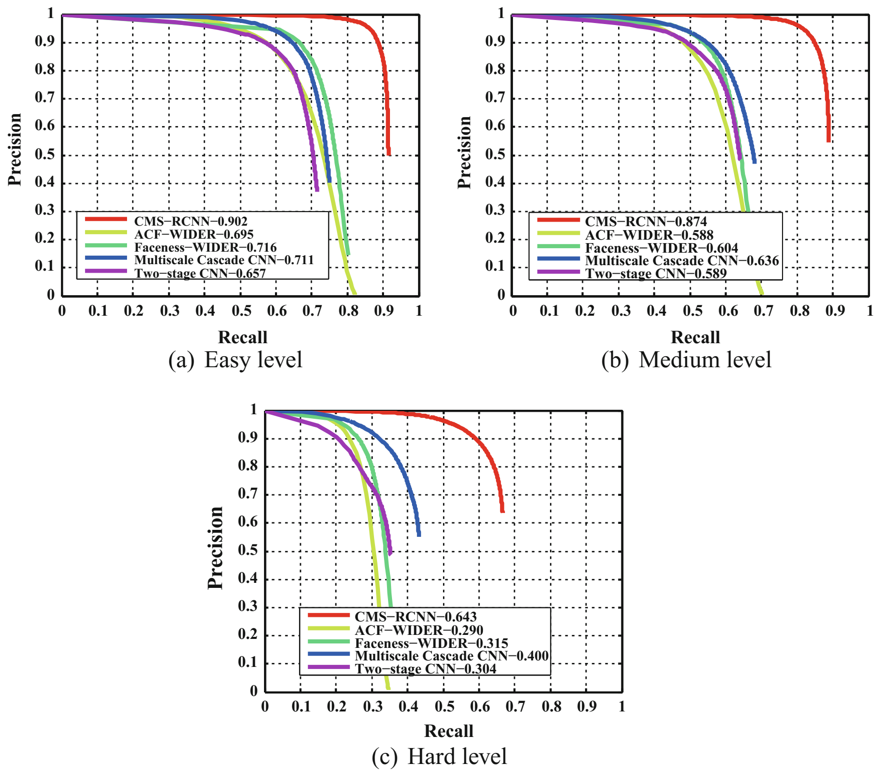


Fig. 3.7 Precision-Recall curves obtained by our proposed CMS-RCNN (*red*) and the other baselines, i.e. Two-stage CNN [1], Multi-scale Cascade CNN [1], Faceness [16], and Aggregate Channel Features (ACF) [11]. All methods trained and tested on the same training and testing set of the WIDER FACE dataset. **a** Easy level, **b** Medium level, and **c** Hard level. Our method achieves the state-of-the-art results with the highest AP values of 0.902 (Easy), 0.874 (Medium), and 0.643 (Hard) among the methods on this database. It also outperforms the second best baseline by 26.0% (Easy), 37.4% (Medium) and 60.8% (Hard)

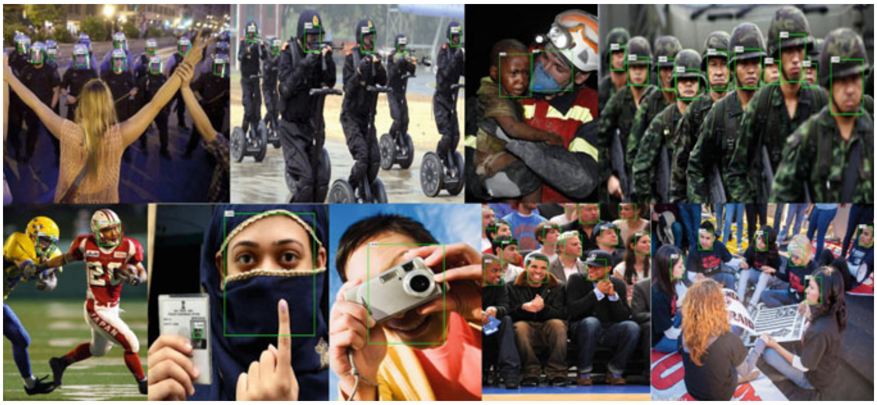


Fig. 3.8 Some examples of face detection results using our proposed CMS-RCNN method on WIDER FACE database [1]

and outperforms the second best baseline by 26.0% (Easy), 37.4% (Medium), and 60.8% (Hard). These results suggest that as the difficulty level goes up, CMS-RCNN can detect challenging faces better. So it has the ability to handle difficult conditions hence is more closed to human detection level. Figure 3.8 shows some examples of face detection results using the proposed CMS-RCNN on this database.

Visualization of False Positives

As it is well known that precision-recall curves get dropped due to the false positives, we are interested in the false positives produced by our CMS-RCNN model. We are curious about what object can fool our model to treat it as a face. Is it due to over-fitting, data bias, or miss labeling?

In order to visualize the false positives, we test the CMS-RCNN model on the WIDER FACE validation set and pick all the false positives according to the ground truth. Then those positives are sorted by the confidence score in a descending order. We choose the top 20 false positives as illustrated in Fig. 3.9. Because their confidence scores are high, they are the objects most likely to cause our model making mistakes. It turns out that most of the false positives are actually human faces caused by miss labeling, which is a problem of the dataset itself. For other false positives, we find the errors made by our model are rather reasonable. They all have the pattern of human face as well as the shape of human body.

3.5.2 Experiments on FDDB Face Database

To show that our method generalizes well to other database, the proposed CMS-RCNN is also benchmarked on the FDDB database [20]. It is a standard database for testing and evaluation of face detection algorithms. It contains annotations for

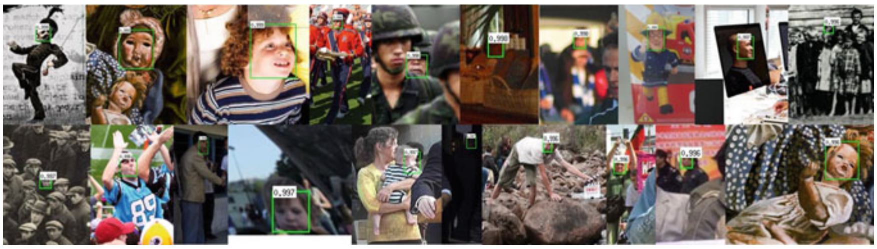


Fig. 3.9 Examples of the top 20 false positives from our CMS-RCNN model tested on the WIDER FACE validation set. In fact these false positives include many human faces not in the dataset due to mislabeling, which means that our method is robust to the noise in the data

5,171 faces in a set of 2,845 images taken from the Faces in the Wild dataset. Most of the images in the FDDB database contain less than three faces that are clear or slightly occluded. The faces generally have large sizes and high resolutions compared to WIDER FACE. We use the same model trained on WIDER FACE training set presented in Sect. 3.5.1 to perform the evaluation on the FDDB database.

The evaluation is performed based on the discrete criterion following the same rules in PASCAL VOC Challenge [37], i.e., if the ratio of the intersection of a detected region with an annotated face region is greater than 0.5, it is considered as a true positive detection. The evaluation is proceeded following the FDDB evaluation protocol and compared against the published methods provided in the protocol, i.e. HyperFace [19], DP2MFD [17], CCF [18], Faceness [16], NPDFace [13], MultiresHPM [12], DDFD [15], CascadeCNN [14], ACF-multiscale [11], Pico [7], HeadHunter

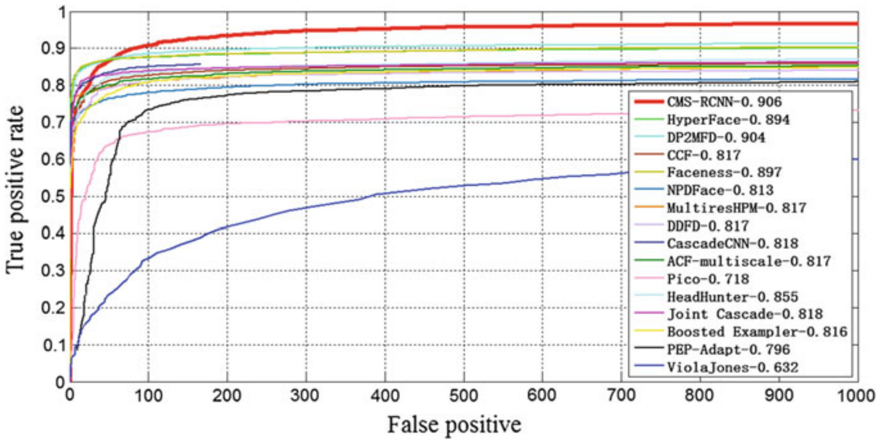


Fig. 3.10 ROC curves of our proposed CMS-RCNN and the other published methods on FDDB database [20]. Our method achieves the best recall rate on this database. Numbers in the legend show the average precision scores

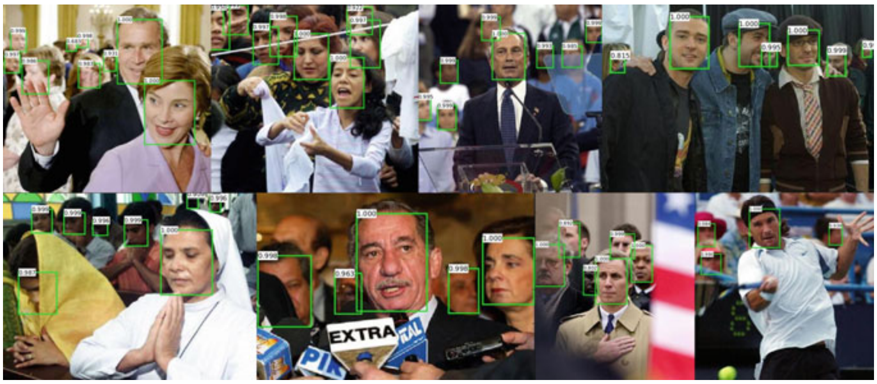


Fig. 3.11 Some examples of face detection results using our proposed CMS-RCNN method on FDDB database [20]

[9], Joint Cascade [10], Boosted Exemplar [8], and PEP-Adapt [6]. The proposed CMS-RCNN approach outperforms most of the published face detection methods and achieves a very high recall rate comparing against all other methods (as shown Fig. 3.10). This is concrete evidence to demonstrate that CMS-RCNN robustly detects unconstrained faces. Figure 3.11 shows some examples of the face detection results using the proposed CMS-RCNN on the FDDB dataset.

3.6 Conclusion and Future Work

This paper has presented our proposed CMS-RCNN approach to robustly detect human facial regions from images collected under various challenging conditions, e.g., highly occlusions, low resolutions, facial expressions, illumination variations, etc. The approach is benchmarked on two challenging face detection databases, i.e., the WIDER FACE Dataset and the FDDB, and compared against recent other face detection methods. The experimental results show that our proposed approach outperforms strong baselines on the WIDER FACE and consistently achieves very competitive results against state-of-the-art methods on the FDDB.

In our implementation, the proposed CMS-RCNN consists of the MS-RPN and the CMS-CNN. During training, they are merged together in an approximate joint training style for each SGD iteration, in which the derivatives w.r.t. the proposal boxes' coordinates are ignored. In the future, we want to go to the fully joint training so that the network can be trained in end-to-end fashion (Fig. 3.12).

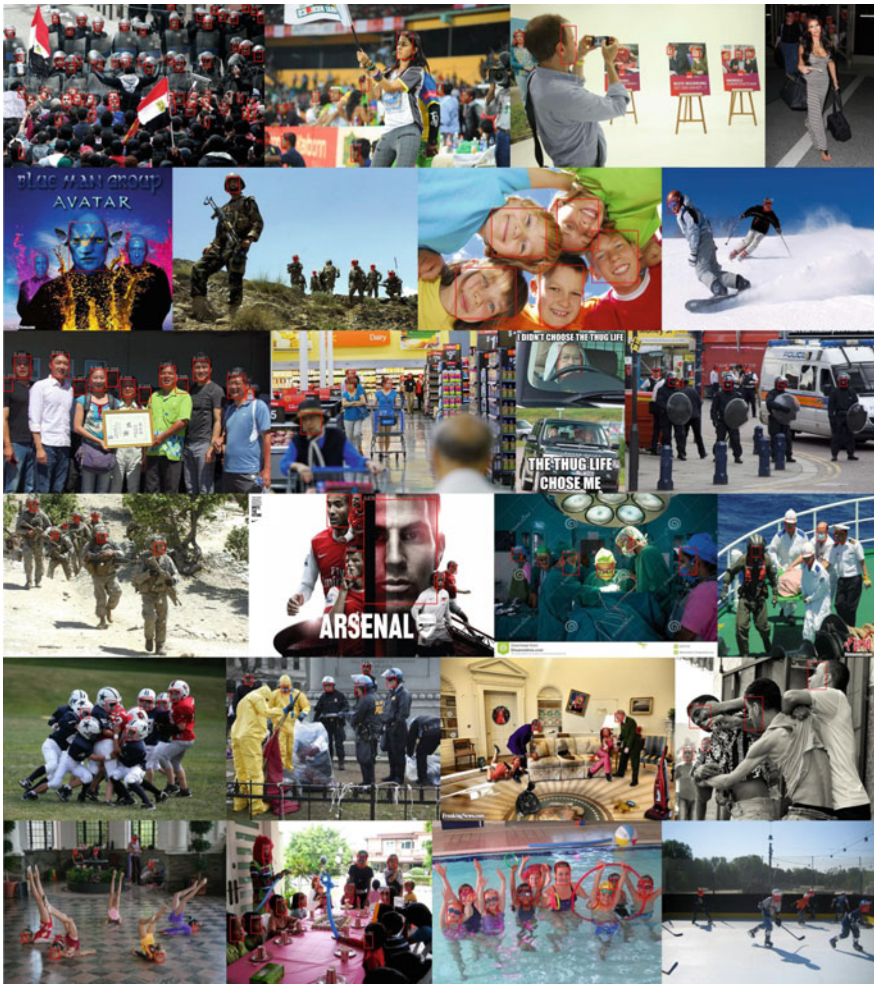


Fig. 3.12 More results of unconstrained face detection under challenging conditions using our proposed CMS-RCNN

References

1. S. Yang, P. Luo, C.C. Loy, X. Tang, Wider face: a face detection benchmark, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2016), pp. 5525–5533
2. P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2001. CVPR 2001*, vol. 1 (IEEE, 2001), pp. I–511
3. C. Zhang, Z. Zhang, A survey of recent advances in face detection. Technical Report MSR-TR-2010-66 (2010), http://research.microsoft.com/apps/pubs/default.aspx?id=132077

4. X. Zhu, D. Ramanan, Face detection, pose estimation, and landmark localization in the wild, in *2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2012), pp. 2879–2886
5. J. Li, Y. Zhang, Learning surf cascade for fast and accurate object detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 3468–3475
6. H. Li, G. Hua, Z. Lin, J. Brandt, J. Yang, Probabilistic elastic part model for unsupervised face detector adaptation, in *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 793–800
7. N. Markuš, M. Frljak, I.S. Pandžić, J. Ahlberg, R. Forchheimer, A method for object detection based on pixel intensity comparisons organized in decision trees (2013). arXiv:1305.4537
8. H. Li, Z. Lin, J. Brandt, X. Shen, G. Hua, Efficient boosted exemplar-based face detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1843–1850
9. M. Mathias, R. Benenson, M. Pedersoli, L. Van Gool, Face detection without bells and whistles, in *Computer Vision-ECCV 2014* (Springer, Berlin, 2014), pp. 720–735)
10. D. Chen, S. Ren, Y. Wei, X. Cao, J. Sun, Joint cascade face detection and alignment, in *Computer Vision-ECCV 2014* (Springer, Berlin, 2014), pp. 109–122
11. B. Yang, J. Yan, Z. Lei, S.Z. Li, Aggregate channel features for multi-view face detection, in *2014 IEEE International Joint Conference on Biometrics (IJCB)* (IEEE, 2014), pp. 1–8
12. G. Ghiasi, C.C. Fowlkes, Occlusion coherence: detecting and localizing occluded faces (2015). arXiv:1506.08347
13. S. Liao, A. Jain, S. Li, A fast and accurate unconstrained face detector (2014)
14. H. Li, Z. Lin, X. Shen, J. Brandt, G. Hua, A convolutional neural network cascade for face detection, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 5325–5334
15. S.S. Farfade, M.J. Saberian, L.-J. Li, Multi-view face detection using deep convolutional neural networks, in *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval* (ACM, 2015), pp. 643–650
16. S. Yang, P. Luo, C.-C. Loy, X. Tang, From facial parts responses to face detection: a deep learning approach, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 3676–3684
17. R. Ranjan, V.M. Patel, R. Chellappa, A deep pyramid deformable part model for face detection, in *2015 IEEE 7th International Conference on Biometrics Theory, Applications and Systems (BTAS)* (IEEE, 2015), pp. 1–8
18. B. Yang, J. Yan, Z. Lei, S.Z. Li, Convolutional channel features, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 82–90
19. R. Ranjan, V.M. Patel, R. Chellappa, Hyperface: a deep multi-task learning framework for face detection, landmark localization, pose estimation, and gender recognition (2016). arXiv:1603.01249
20. V. Jain, E. Learned-Miller, FDDB: a benchmark for face detection in unconstrained settings. University of Massachusetts, Amherst, Technical Report UM-CS-2010-009 (2010)
21. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014). arXiv:1409.1556
22. P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models. IEEE Trans. PAMI **32**(9), 1627–1645 (2010)
23. X. Yu, J. Huang, S. Zhang, W. Yan, D. Metaxas, Pose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model, in *Proceedings of the IEEE International Conference on Computer Vision* (2013), pp. 1944–1951
24. S.K. Divvala, D. Hoiem, J.H. Hays, A.A. Efros, M. Hebert, An empirical study of context in object detection, in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2009* (IEEE, 2009), pp. 1271–1278
25. S. Bell, C.L. Zitnick, K. Bala, R. Girshick, Inside-outside net: detecting objects in context with skip pooling and recurrent neural networks (2015). arXiv:1512.04143

26. S. Zagoruyko, A. Lerer, T.-Y. Lin, P.O. Pinheiro, S. Gross, S. Chintala, P. Dollár, A multipath network for object detection (2016). arXiv:1604.02135
27. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
28. R. Girshick, J. Donahue, T. Darrell, J. Malik, Region-based convolutional networks for accurate object detection and segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **38**(1), 142–158 (2016)
29. R. Girshick, Fast R-CNN, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1440–1448
30. S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, in *Advances in Neural Information Processing Systems* (2015), pp. 91–99
31. M.D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in *Computer Vision-ECCV 2014* (Springer, Berlin, 2014), pp. 818–833
32. T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in *ECCV* (2014), pp. 740–755
33. B. Hariharan, P. Arbeláez, R. Girshick, J. Malik, Hypercolumns for object segmentation and fine-grained localization, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 447–456
34. W. Liu, A. Rabinovich, A.C. Berg, Parsenet: looking wider to see better (2015). arXiv:1506.04579
35. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, in *Proceedings of the ACM International Conference on Multimedia* (ACM, 2014), pp. 675–678
36. C.L. Zitnick, P. Dollár, Edge boxes: locating object proposals from edges, in *ECCV* (Springer, Berlin, 2014), pp. 391–405
37. M. Everingham, L. Van Gool, C.K. Williams, J. Winn, A. Zisserman, The pascal visual object classes (VOC) challenge. Int. J. Comput. Vis. **88**(2), 303–338 (2010)

Part II
Deep Learning for Fingerprint, Fingervein and Iris Recognition

Chapter 4
Latent Fingerprint Image Segmentation Using Deep Neural Network

Jude Ezeobiejesi and Bir Bhanu

Abstract We present a deep artificial neural network (DANN) model that learns latent fingerprint image patches using a stack of restricted Boltzmann machines (RBMs), and uses it to perform segmentation of latent fingerprint images. Artificial neural networks (ANN) are biologically inspired architectures that produce hierarchies of maps through learned weights or filters. Latent fingerprints are fingerprint impressions unintentionally left on surfaces at a crime scene. To make identifications or exclusions of suspects, latent fingerprint examiners analyze and compare latent fingerprints to known fingerprints of individuals. Due to the poor quality and often complex image background and overlapping patterns characteristic of latent fingerprint images, separating the fingerprint region of interest from complex image background and overlapping patterns is very challenging. Our proposed DANN model based on RBMs learns fingerprint image patches in two phases. The first phase (unsupervised pre-training) involves learning an identity mapping of the input image patches. In the second phase, fine-tuning and gradient updates are performed to minimize the cost function on the training dataset. The resulting trained model is used to classify the image patches into fingerprint and non-fingerprint classes. We use the fingerprint patches to reconstruct the latent fingerprint image and discard the non-fingerprint patches which contain the structured noise in the original latent fingerprint. The proposed model is evaluated by comparing the results from the state-of-the-art latent fingerprint segmentation models. The results of our evaluation show the superior performance of the proposed method.

J. Ezeobiejesi (✉) · B. Bhanu
Center for Research in Intelligent Systems, University of California at Riverside, Riverside, CA 92521, USA
e-mail: jezeobie@cs.ucr.edu

B. Bhanu
e-mail: bhanu@cris.ucr.edu

© Springer International Publishing AG 2017
B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*, Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_4

4.1 Introduction

Deep learning is a technique for learning features using hierarchical layers of neural networks. There are usually two phases in deep learning. The first phase commonly referred to as pre-training involves unsupervised, layer-wise training. The second phase (fine-tuning) involves supervised training that exploits the results of the first phase. In deep learning, hierarchical layers of learned abstraction are used to accomplish high level tasks [3]. In recent years, deep learning techniques have been applied to a wide variety of problems in different domains [3]. Some of the notable areas that have benefited from deep learning include pattern recognition [21], computer vision [16], natural language processing, and medical image segmentation [17]. In many of these domains, deep learning algorithms outperformed previous state-of-the-art algorithms.

Latent fingerprints are fingerprint impressions unintentionally left on surfaces at a crime scene. Latent examiners analyze and compare latent fingerprints to known fingerprints of individuals to make identifications or exclusions of suspects [9]. Reliable latent fingerprint segmentation is an important step in the automation of latent fingerprint processing. Better latent fingerprint matching results can be achieved by having automatic latent fingerprint segmentation with a high degree of accuracy. In recent years, the accuracy of latent fingerprint identification by latent fingerprint forensic examiners has been the subject of increased study, scrutiny, and commentary in the legal system and the forensic science literature. Errors in latent fingerprint matching can be devastating, resulting in missed opportunities to apprehend criminals or wrongful convictions of innocent people. Several high-profile cases in the United States and abroad have shown that forensic examiners can sometimes make mistakes when analyzing or comparing fingerprints [14] manually. Latent fingerprints have significantly poor quality ridge structure and large nonlinear distortions compared to rolled and plain fingerprints. As shown in Fig. 4.1, latent fingerprint images contain background structured noise such as stains, lines, arcs, and sometimes text. The poor quality and often complex image background and overlapping patterns characteristic of latent fingerprint images make it very challenging to separate the fingerprint regions of interest from complex image background and overlapping patterns [29]. To process latent fingerprints, latent experts manually mark the regions of interest (*ROIs*) in latent fingerprints and use the *ROIs* to search large databases of reference full fingerprints and identify a small number of potential matches for manual examination. Given the large size of law enforcement databases containing rolled and plain fingerprints, it is very desirable to perform latent fingerprint processing in a fully automated way. As a step in this direction, this chapter proposes an efficient technique for separating latent fingerprints from the complex image background using deep learning. We learn a set of features using a hierarchy of RBMs. These features are then passed to a supervised learning algorithm to learn a classifier for patch classification. We use the result of the classification for latent fingerprint image segmentation. To the best of our knowledge, no previous work has used this strategy to segment latent fingerprints.

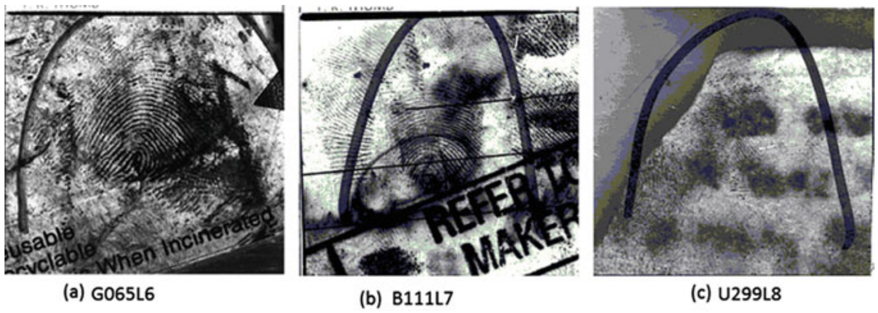


Fig. 4.1 Sample latent fingerprints from NIST SD27 showing three different quality levels **a** good, **b** bad, and **c** ugly

The rest of the chapter is organized as follows: Sect. 4.2.1 reviews recent works in latent fingerprint segmentation while Sect. 4.2.1.1 describes the contributions of this chapter. Section 4.3 highlights our technical approach and presents an overview of RBM as well discussion on learning with RBMs. The experimental results and performance evaluation of our proposed approach are presented in Sect. 4.4. Section 4.4.5 highlights the impacts of diffusing the training dataset with fractal dimension and lacunarity features on the performance of the network while Sect. 4.5 contains the conclusions and future work.

4.2 Related Work and Contributions

4.2.1 Related Work

Recent studies carried out on latent fingerprint segmentation can be grouped into three categories:

- Techniques based on classification of image patches
- Techniques based on clustering
- Techniques that rely on ridge frequency and orientation properties

The study presented in [9] falls into the first category. The authors performed image segmentation by extracting 8×8 nonoverlapping patches from a latent fingerprint image and classifying them into fingerprint and non-fingerprint patches using fractal dimension features computed for each image patch. They assembled the fingerprint patches to build the fingerprint portion (segmented region of interest) of the original image.

In the second category of approaches, Arshad et al. [4] used K-means clustering to divide the latent fingerprint image into nonoverlapping blocks and computed the standard deviation of each block. They considered a block as foreground if its standard deviation is greater than a predefined threshold otherwise, it was a background block. They used morphological operations to segment the latent fingerprint.

The approaches that fall into the third category rely on the analysis of the ridge frequency and orientation properties of the ridge valley patterns to determine the area within a latent fingerprint image that contains the fingerprint [4, 7, 13, 27, 29]. Choi et al. [7] used orientation tensor approach to extract the symmetric patterns of a fingerprint and removed the structural noise in background. They used a local Fourier analysis method to estimate the local frequency in the latent fingerprint image and located fingerprint regions by considering valid frequency ranges. They obtained candidate fingerprint (foreground) regions for each feature (orientation and frequency) and then localized the latent fingerprint regions using the intersection of those candidate regions. Karimi et al. [13] estimated local frequency of the ridge/valley pattern based on ridge projection with varying orientations. They used the variance of frequency and amplitude of ridge signal as features for the segmentation algorithm. They reported segmentation results for only two latent fingerprint images and provided no performance evaluation. Short et al. [27] proposed the ridge template correlation method for latent fingerprint segmentation. They generated an ideal ridge template and computed cross-correlation value to define the local fingerprint quality. They manually selected six different threshold values to assign a quality value to each fingerprint block. They neither provided the size and number for the ideal ridge template nor reported evaluation criteria for the segmentation results. Zhang et al. [29] proposed an adaptive total variation (TV) model for latent fingerprint segmentation. They adaptively determined the weight assigned to the fidelity term in the model based on the background noise level. They used it to remove the background noise in latent fingerprint images.

Our approach uses a deep architecture that performs learning and classification in a two-phase approach. The first phase (unsupervised pre-training) involves learning an identity mapping of the input image patches. In the second phase (fine-tuning), the model performs gradient updates to minimize the cost function on the dataset. The trained model is used to classify the image patches and the results of the classification are used for latent fingerprint image segmentation.

4.2.1.1 Contributions

This chapter makes the following contributions:

- Modification of how the standard RBM learning algorithm is carried out to incorporate a weighting scheme that enables the RBM in the first layer to model the input data with near zero reconstruction error. This enabled the higher level weights to model the higher level data efficiently.
- A cost function based on weighted harmonic mean of missed detection rate and false detection rate is introduced to make the network learn the minority class better, and improve per class accuracy. By heavily penalizing the misclassication of minority (fingerprint) class, the learned model is tuned to achieve close to zero missed detection rate for the minority class.

- The proposed generative feature learning model and associated classifier yield state-of-the-art performance on latent fingerprint image segmentation that is consistent across many latent fingerprint image databases.

4.3 Technical Approach

Our approach involves partitioning a latent fingerprint image into 8×8 nonoverlapping blocks, and learning a set of stochastic features that model a distribution over image patches using a generative multilayer feature extractor. We use the features to train a single-layer perceptron classifier that classifies the patches into fingerprint and non-fingerprint classes. We use the fingerprint patches to reconstruct the latent fingerprint image and discard the non-fingerprint patches which contain the structured noise in the original latent fingerprint. The block diagram of our proposed approach is shown in Fig. 4.2, and the architecture of the feature learning, extraction, and classification model is shown in Fig. 4.3.

4.3.1 Restricted Boltzmann Machine

A restricted Boltzmann machine is a stochastic neural network that consists of visible layer, hidden layer, and a bias unit [11]. A sample RBM with binary visible and hidden units is shown in Fig. 4.4. The energy function E_f of RBM is linear in its free parameters and is defined as [11]:

$$E_f(\hat{x}, h) = -\sum_i b_i \hat{x}_i - \sum_j c_j h_j - \sum_i \sum_j \hat{x}_i w_{i,j} h_j, \tag{4.1}$$

where $\hat{x}$ and h represent the visible and hidden units, respectively, W represents the weights connecting $\hat{x}$ and h, while b and c are biases of the visible and hidden units, respectively. The probability distributions over visible or hidden vectors are defined in terms of the energy function [11]:

$$P(\hat{x}, h) = \frac{1}{\omega} e^{-E_f(\hat{x},h)}, \tag{4.2}$$

where ω is a partition function that ensures the probability distribution of over all possible configurations of the hidden or visible vectors sum to 1. The marginal probability of a visible vector $P(\hat{x})$ is the sum over all possible hidden layer configurations [11] and is defined as:

$$P(\hat{x}) = \frac{1}{\omega} \sum_h e^{-E_f(\hat{x},h)} \tag{4.3}$$

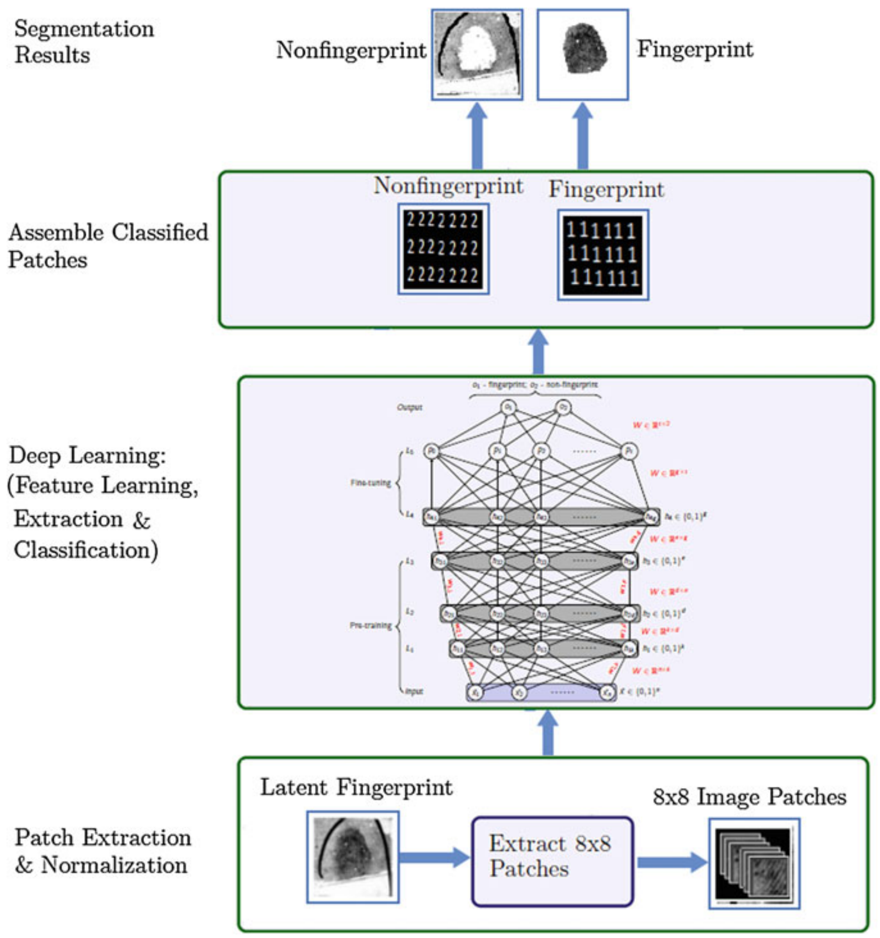


Fig. 4.2 Proposed approach

RBM has no intra-layer connections and given the visible unit activations, the hidden unit activations are mutually independent. Also the visible unit activations are mutually independent given the hidden unit activations [6]. The conditional probability of a configuration of the visible units is given by

$$P(\hat{x}|h) = \prod_{i=1}^{n} P(\hat{x}_i|h), \tag{4.4}$$

where n is the number of visible units. The conditional probability of a configuration of hidden units given visible units is

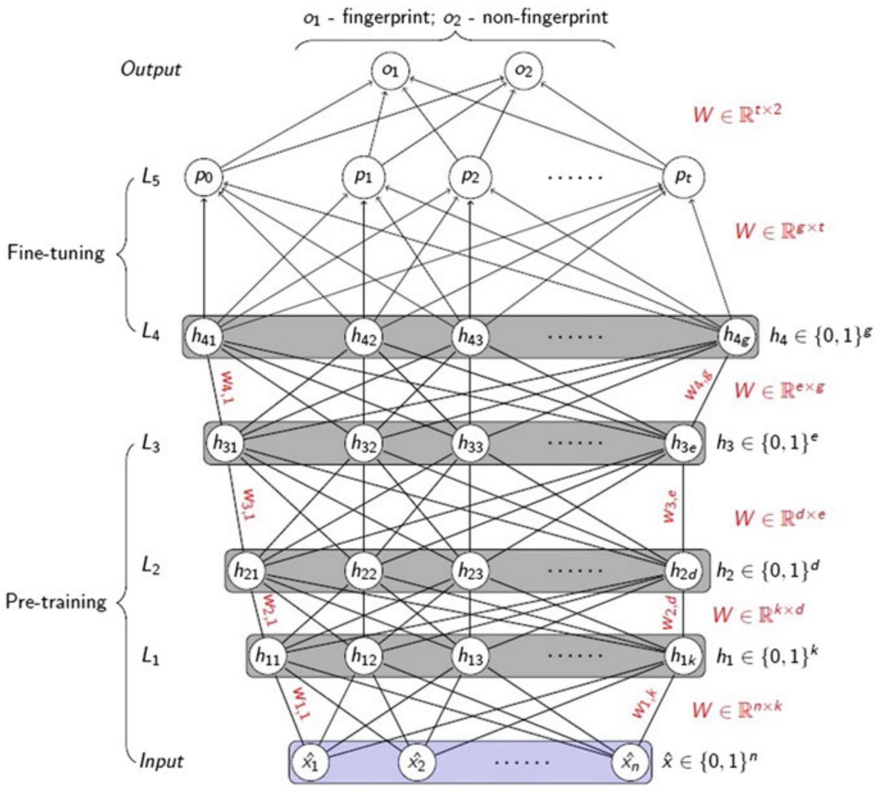


Fig. 4.3 Feature learning, extraction, and classification using a multilayer neural network. The pre-training phase uses the input layer (visible units), and three hidden layers of RBM (L_1, L_2, L_3). The fine-tuning phase uses an RBM layer (L_4) and a single-layer perceptron (L_5). The output layer has two output neurons (fingerprint and non-fingerprint). All the units are binary. $h_{i,j}$ is the jth node in L_i, $w_{i,j}$ is the weight connecting the ith node in layer L_i to the jth node in layer L_{i-1}. We set $n = 81$ (64 from 8×8 and 17 from diffusion), $k = 800$, $d = 1000$, $e = 1200$, $g = 1200$, $t = 1200$, where n, k, d, e, g, t are the number of nodes in the input layer, L_1, L_2, L_3, L_4, L_5, respectively

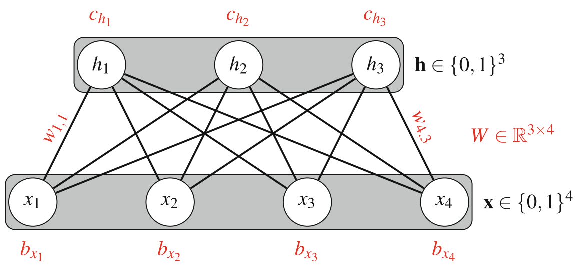


Fig. 4.4 Graphical depiction of RBM with binary visible and hidden units. $x_i, i = 1, \ldots, 4$, are the visible units while $h_k, k = 1, \ldots, 3$, are the hidden units. $b_{x_i}, i = 1, \ldots, 4$, are the biases for the visible units and $c_{h_k}, k = 1, \ldots, 3$, are the biases for the hidden units

$$P(h|\hat{x}) = \prod_{j=1}^{m} P(h_j|\hat{x}), \tag{4.5}$$

where m is the number of hidden units. The individual activation probabilities are given by

$$P(h_j = 1|\hat{x}) = \sigma\left(b_j + \sum_{i=1}^{n} w_{i,j}\hat{x}_i\right) \tag{4.6}$$

and

$$P(\hat{x}_i = 1|h) = \sigma\left(c_i + \sum_{j=1}^{m} w_{i,j}h_j\right), \tag{4.7}$$

where c_i is the ith hidden unit bias, b_j is the jth visible unit bias, $w_{i,j}$ is the weight connecting the ith visible unit and jth hidden unit, and σ is the logistic sigmoid.

4.3.2 Learning with RBM

Learning with RBM involves several steps of sampling hidden variables given visible variables, sampling visible variables given hidden variables, and minimizing reconstruction error by adjusting the weights between the hidden unit and visible layers. The goal of learning with RBM is to identify the relationship between the hidden and visible variables using a process akin to identity mapping. We performed the sampling step using Gibbs sampling technique enhanced with contrastive divergence.

4.3.2.1 Gibbs Sampling

A sampling algorithm based on Monte Carlo Markov Chain (MCMC) technique used in estimating desired expectations in learning models. It allows for the computation of statistics of a posterior distribution of given simulated samples from that distribution [28]. A Gibbs sampling of the joint of R random variables $R = (R_1, R_2, \ldots, R_n)$ involves a sequence of R sampling sub-steps of the form $R_i \sim p(R_i|R_{-i})$ where R_i contains the *n-1* other random variables in R excluding R_i. For RBMs, $R = Q_1 \cup Q_1$ where $Q_1 = \{\hat{x}_i\}$ and $Q_2 = \{h_i\}$. Given that the sets Q_1 and Q_2 are conditionally independent, the visible units can be sampled simultaneously given fixed values of the hidden units using block Gibbs sampling. Similarly, the hidden units can be sampled simultaneously given the visible units. The following is a step in the Markov chain:
$h^{(k+1)} \sim \sigma(W^T v^{(k)} + c)$
$\hat{x}^{(k+1)} \sim \sigma(Wh^{(k+1)} + b)$,

where $h^{(k)}$ refers to the set of all hidden units at the kth step of the Markov chain and σ denotes logistic sigmoid defined as

$$o(x) = \frac{1}{1 + e^{-W_v z(x) - b}} \tag{4.8}$$

$$\text{with } z(x) = \frac{1}{1 + e^{-W_h x - c}}, \tag{4.9}$$

where W_h and c are the weight matrix and bias for the hidden layers excluding the first layer, and $z(x)$ is the activation of the hidden layer in the network. W_v is a weight matrix connecting the visible layer to the first hidden layer, and b is a bias for the visible layer.

4.3.2.2 Contrastive Divergence (CD-k)

This algorithm is used inside gradient descent procedure to speed up Gibbs sampling. It helps in optimizing the weight W during RBM training. CD-k speeds up Gibbs sampling by taking sample after only k-steps of Gibbs sampling, without waiting for the convergence of the Markov chain. In our experiments we set $k = 1$.

4.3.2.3 Stochastic Gradient Descent

With large datasets, computing the cost and gradient for the entire training set is usually very slow and may be intractable [24]. This problem is solved by Stochastic Gradient Descent (SGD) by following the negative gradient of the objective function after seeing a few training examples. SGD is used in neural networks to mitigate the high cost of running backpropagation over the entire training set [24].

Given an objective function $J(\phi)$, the standard gradient descent algorithm updates the parameters ϕ as follows:

$$\phi = \phi - \alpha \nabla_\phi E[J(\phi], \tag{4.10}$$

where the expectation $E[J(\phi]$ is obtained through an expensive process of evaluating the cost and gradient over the entire training set. With SGD, the gradient of the parameters are computed using a few training examples with no expectation to worry about. The parameters are update as,

$$\phi = \phi - \alpha \nabla_\phi J(\phi; x^{(i)}, y^{(i)}) \tag{4.11}$$

where the pair $(x^{(i)}, y^{(i)})$ are from the training set. Each parameter update is computed using a few training examples. This reduces the variance in the parameter update with the potential of leading to more stable convergence. Prior to each training epoch, we randomly shuffled the training data to avoid biasing the gradient. Presenting the training data to the network in a nonrandom order could bias the gradient and lead to poor convergence.

One of the issues with learning with stochastic gradient descent is the tendency of the gradients to decrease as they are backpropagated through multiple layer of nonlinearity. We worked around this problem by using different learning rates for each layer in the proposed network.

4.3.2.4 Cost Function

Our goal is to classify all fingerprint patches (minority class) correctly to meet our segmentation objective of extracting the region of interest (fingerprint part) from the latent fingerprint image. We introduced a cost function based on the weighted harmonic mean of missed detection rate and false detection rate. We adopted a weight assignment scheme that was skewed in favor of the minority class to make the neural network learn the minority class better. Given a set of weights $w_1, w_2, \ldots, w_n$ associated with a dataset $x_1, x_2, \ldots, x_n$, the weighted harmonic mean $\mathbb{H}$ is defined as

$$\mathbb{H} = \frac{\sum_{i=1}^{n} w_i}{\sum_{i=1}^{n} \frac{w_i}{x_i}} = \left(\frac{\sum_{i=1}^{n} w_i x_i^{-1}}{\sum_{i=1}^{n} w_i} \right)^{-1}. \tag{4.12}$$

By penalizing the misclassification of minority class more, the model learned to detect minority class with a high degree of accuracy. The cost function is defined as:

$$\mathbb{C} = \frac{2}{\frac{1}{\tau MDR} + \frac{1}{\tau FDR}}, \tag{4.13}$$

where τMDR and τFDR are the weighted missed detection rate and weighted false detection rate, respectively. They are computed as: $\tau MDR = \tau_1 * MDR$ and $\tau FDR = \tau_2 * FDR$, where $\tau_1 = \frac{P_s + N_s}{P_s}$ and $\tau_2 = \frac{P_s + N_s}{N_s}$ are the weights assigned to positive class samples P_s and negative class samples N_s, respectively.

Table 4.1 shows a comparison of the error cost during the fine-tuning phase of our model with cross entropy cost function, and the proposed cost function.

4.3.3 Choice of Hyperparameters

We selected the value of the hyperparameters used in the proposed network based on the performance of the network on the validation set. The parameters and their values are shown in Table 4.2.

Table 4.1 Comparison of model performance using regular cost function (cross entropy) and proposed cost function. The mean, maximum, and minimum error costs are better (smaller is better) with the proposed cost function. With the proposed cost function, the model is tuned to achieve a low missed detection rate

Cost function	Min. error cost	Max. error cost	Mean error cost
Cross entropy	3.53E-03	1.041E+00	6.29E-02
Proposed	6.00E-04	1.10E-02	2.03E-03

Table 4.2 Parameters and values

Parameter	L_0	L_1	L_2	L_3	L_4	L_5	L_6
Number of neurons	81	800	1000	1200	1200	1200	2
Batch size	–	100	100	100	100	100	–
Epochs	–	50	50	50	50	–	–
Learning rate	–	1e-3	5e-4	5e-4	5e-4	–	–
Momentum	–	0.70	0.70	0.70	0.70	–	
Number of iterations	–	–	–	–	–	50	–

4.3.3.1 Unsupervised Pre-training

We adopt unsupervised layer-wise pre-training because of its power in capturing the dominant and statistically reliable features present in the dataset. The output of each layer is a representation of the input data embodying those features. According to [8], greedy layer-wise unsupervised pre-training overcomes the challenges of deep learning by introducing a useful prior to the supervised fine-tuning training procedure. After pre-training a layer, its input sample is reconstructed and the mean square reconstruction error is computed. The reconstruction step entails guessing the probability distribution of the original input sample in a process referred to as generative learning. Unsupervised pre-training promotes input transformations that capture the main variations in the dataset distribution [8]. Since there is a possibility that only a small subset of these variations may be relevant for predicting the class label of a sample, using a small number of nodes in the hidden layers will make it less likely for the transformations necessary for predicting the class label to be included in the set of transformations learned by unsupervised pre-training. This idea is reflected in our choice of the number of nodes in the pre-training layers. We ran several experiments to determine the optimal nodes in each of the three pre-training layers. As shown in Table 4.2, the number of nodes in the pre-training layers L_1, L_2, and L_3 are 800, 1000, and 1200, respectively.

4.3.3.2 Supervised Fine-Tuning

Supervised fine-tuning is the process of backpropagating the gradient of a classifier's cost through the feature extraction layers. Supervised fine-tuning boosts the performance of neural networks with unsupervised pre-training [19]. In our model, supervised fine-tuning is done with a layer of RBM and a single-layer perceptron depicted as L_4 and L_5, respectively in Fig. 4.3. During the fine-tuning phase, we initialized L_4, with the pre-trained weights of the top-most pre-training layer L_3.

4.4 Experiments and Results

We implemented our algorithms in MATLAB R2014a running on Intel Core i7 CPU with 8GB RAM and 750GB hard drive. Our implementation relied on NNBox, a MATLAB toolbox for neural networks. The implementation uses backpropagation, contrastive divergence, Gibbs sampling, and hidden units sparsity based optimization techniques.

4.4.1 Latent Fingerprint Databases

We tested our model on the following databases:

• **NIST SD27:** This database was acquired from the National Institute of Standards and Technology. It contains images of **258** latent crime scene fingerprints and their matching rolled tenprints. The images in the database are classified as good, bad, or ugly based on the quality of the image. The latent prints and rolled prints are at **500** ppi.

• **WVU Database:** This database is jointly owned by West Virginia University and the FBI. It has **449** latent fingerprint images and matching **449** rolled fingerprints. All images in this database are at **1000** ppi.

• **IIITD Database:** The IIITD was obtained from the Image Analysis and Biometrics lab at the Indraprastha Institute of Information Technology, Delhi, India [25]. There are **150** latent fingerprints and **1,046** exemplar fingerprints. Some of the fingerprint images are at **500** ppi while others are at **1000** ppi.

4.4.2 Performance Evaluation and Metrics

We used the following metrics to evaluate the performance of our network.

- **Missed Detection Rate** (***MDR***): This is the percentage of fingerprint patches classified as non-fingerprint patches and is defined as.

$$MDR = \frac{FN}{TP + FN} \tag{4.14}$$

where *FN* is the number of false negatives and *TP* is the number of true positives.

- **False Detection Rate** (***FDR***): This is the percentage of non-fingerprint patches classified as fingerprint patches.It is defined as

$$FDR = \frac{FP}{TN + FP} \tag{4.15}$$

where *FP* is the number of false positives and *TN* is the number of true negatives.

- **Segmentation Accuracy (SA):** It gives a good indication of the segmentation reliability of the model.

$$SA = \frac{TP + TN}{TP + FN + TN + FP} \tag{4.16}$$

4.4.3 Stability of the Architecture

To investigate the stability of the proposed architecture, we performed five runs of training the network using 50,000 training samples. All the model parameters (number of epochs, number of iterations etc.) shown in Table 4.2 remained unchanged across the runs. The mean square reconstruction error (msre), mean error cost, and standard deviation for the five runs are shown in Table 4.3. Plots of the reconstruction errors against number of training epochs as well as that of error cost against number or iterations during each run are shown in Fig. 4.5. These results show that our model is stable.

Table 4.3 Network Stability: The msre, error cost, MDR, FDR, and training accuracy for the five different runs are close. The mean and standard deviation indicate stability across the five runs

Run #	MSRE	Error cost	MDR	FDR	Training accuracy
1	0.0179	5.469e-04	2.010e-04	0.00	4.00e-05
2	0.0183	5.406e-04	3.020e-04	0.00	6.00e-05
3	0.0178	5.560e-04	1.010e-04	0.00	2.00e-05
4	0.0179	5.438e-04	2.020e-04	0.00	5.00e-05
5	0.0178	6.045e-04	1.010e-04	0.00	2.00e-05
Mean	0.0179	5.584e-04	1.814e-04	0.00	3.800e-05
Standard deviation	0.0002	2.643e-05	8.409e-05	0.00	1.789e-05

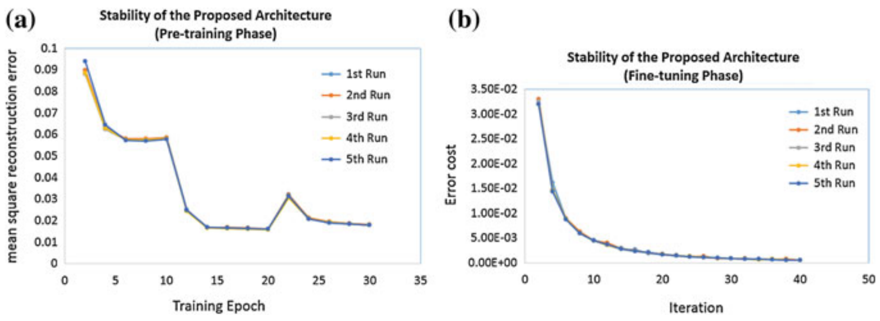


Fig. 4.5 Network Stability: **a** shows that the mean square reconstruction error (msre) followed the same trajectory during the five different runs, converging close to 0.02% msre. Similarly, **b** shows that the error cost during the fine-tuning phase followed the same trajectory for the five runs, converging to about 5.5E-04 error cost. The results are indicative of the stability of the network

4.4.4 Segmentation Using the Trained Network

To segment a latent fingerprint image using the trained network we proceed as follows:

- Split the image into 8×8 nonoverlapping patches and augment each patch data with its fractal dimension and lacunarity features to create a segmentation dataset.
- Normalize the segmentation dataset to have 0 mean and unit standard deviation.
- Load the trained network and compute activation value for each neuron: $a = \sum Wx$
- Feed the activation value to the activation function to normalize it.
- Apply the following thresholding function to obtain the classification results:

$$\theta(x) = \begin{cases} 1 : z > T \\ 0 : z \leq T \end{cases} \tag{4.17}$$

where z is the decision value, x is an example from the segmentation dataset, T is a threshold that gave the best segmentation accuracy on a validation set and was obtained using fivefold cross validation described in Algorithm 1.

4.4.4.1 Searching for Threshold *T*

We implemented a *hook* into the logic of output neurons to access the real-valued output of the activation function. To obtain the percentage of the activation function output for a given neuron, we divided its activation function value by the sum of all activation function values. For each label $y \in 1, 2$, we ordered the validation examples according to their decision values (percentages) and for each pair of adjacent decision values, we checked the segmentation accuracy using their average as T. The algorithm used was inspired by [10], and is shown as Algorithm 1.

Algorithm 1 Searching for threshold T

```
1: procedure THRESHOLD($X, Y$)    ▷ $X$ is the patch dataset, $Y$ is a set of corresponding labels
2:     $num\_class \leftarrow Unique(Y)$    ▷ Get unique labels from $Y$
3:     for $cs = 1, \dots, num\_class$ do    ▷ Iterate over the number of classes
4:         (a) $folds \leftarrow Split(X, 5)$    ▷ Split the validation set into five folds
5:         for $f = 1, \dots, folds$ do    ▷ Iterate over the folds
6:             (i) Run Compute(.) on four folds of validation set    ▷ Run four folds through the
                trained network
7:             (ii)$T_c^f \leftarrow Best()$    ▷ Set $T_c^f$ to the decision value that achieved the best $MDR$
8:         (b) Run Compute(.) on $X$    ▷ Run the validation set through the trained network
9:         $T_c \leftarrow \frac{1}{5}\sum_{k=1}^{folds} T_c^k$    ▷ Set the threshold to the average of the five thresholds from cross
           validation
10:    return $T$    ▷ Return the threshold
```

4.4.5 Dataset Diffusion and Impact on Model Performance

Given a dataset $X = \{x_1, x_2, \dots, x_k\}$, we define the diffusion of X as $\hat{X} = \{x_1, x_2, \dots, x_m\}$, where $m > k$ and each $x_i, k < i < m$ is an element from R^n. In other words, $\hat{X}$ is obtained by *expanding* X with new elements from R^n. A similar idea based on the principle of information diffusion has been used by researchers in situations, where the neural network failed to converge despite adjustments of weights and thresholds [12, 22]. We used features based on fractal dimension and lacunarity to diffuse X. These features help to characterize complex texture in latent fingerprint images [9].

4.4.5.1 Fractal Dimension

Fractal dimension is an index used to characterize texture patterns by quantifying their complexity as a ratio of the change in detail to the change in the scale used. It was defined by Mandelbrot [23] and was first used in texture analysis by Keller et al. [15]. Fractal dimension offers a quantitative way to describe and characterize the complexity of image texture composition [18].

We compute the fractal dimension of an image patch P using a variant of differential box counting (DBC) algorithm [2, 26]. We consider P as a *3-D* spatial surface with *(x,y)* axis as the spatial coordinates and z axis for the gray level of the pixels. Using the same strategy as in *DBC*, we partition the $N \times N$ matrix representing P into nonoverlapping $d \times d$ blocks where $d \in [1, N]$. Each block has a column of boxes of size $d \times d \times h$, where h is the height defined by the relationship $h = \frac{\mathscr{T} d}{N}$, where $\mathscr{T}$ is the total gray levels in P, and d is an integer. Let $\mathscr{T}_{min}$ and $\mathscr{T}_{max}$ be the minimum and maximum gray levels in grid *(i, j)*, respectively. The number of boxes covering block *(i, j)* is given by:

$$n_d(i,j) = floor[\frac{\mathscr{T}_{max} - \mathscr{T}_{min}}{r}] + 1, \tag{4.18}$$

where $r = 2, \ldots, N-1$, is the scaling factor and for each block $r = d$. The number of boxes covering all $d \times d$ blocks is:

$$N_d = \sum_{i,j} n_d(i,j) \tag{4.19}$$

We compute the values N_d for all $d \in [1, N]$. The fractal dimension of each pixel in P is by given by the slope of a plot of the logarithm of the minimum box number as a function of the logarithm of the box size. We obtain a fractal dimension image patch P' represented by an $M \times N$ matrix whose entry *(i, j)* is the fractal dimension FD_{ij} of the pixel at *(i, j)* in *P*.

$$FD_P = \sum_{i=1,j=1}^{MN} FD_{ij} \tag{4.20}$$

⧫ **Fractal Dimension Features**: We implemented a variant of the *DBC* algorithm [2, 26], to compute the following statistical features from the fractal dimension image P'.

• ***Average Fractal Dimension***:

$$FD_{avg} = \frac{1}{MN} \sum_{i=1,j=1}^{MN} FD_{ij} \tag{4.21}$$

• ***Standard Deviation of Fractal Dimension:*** The standard deviation of the gray levels in an image provides a degree of image dispersion and offers a quantitative description of variation in the intensity of the image plane. Therefore

$$FD_{std} = \frac{1}{MN} \sum_{i=1,j=1}^{MN} (FD_{ij} - FD_{avg})^2, \tag{4.22}$$

• ***Fractal Dimension Spatial Frequency:*** This refers to the frequency of change per unit distance across fractal dimension (FD) processed image. We compute it using the formula for (spatial domain) spatial frequency [20]. Given an $N \times N$ FD processed image patch P', let *G(x,y)* be the FD value of the pixel at location *(x,y)* in P'. The row frequency R_f and column frequency C_f are given by

$$R_f = \sqrt{\frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=1}^{N-1} [G(x,y) - G(x,y-1)]^2} \tag{4.23}$$

$$C_f = \sqrt{\frac{1}{MN}\sum_{y=0}^{M-1}\sum_{x=1}^{N-1}[G(x, y) - G(x-1, y)]^2} \tag{4.24}$$

The FD spatial frequency FD_{sf} of P' is defined as

$$FD_{sf} = \sqrt{R_f^2 + C_f^2} \tag{4.25}$$

From signal processing perspective, Eqs. (4.23) and (4.24) favor high frequencies and yield values indicative of patches with fingerprint.

4.4.5.2 Lacunarity

Lacunarity is a second-order statistic that provides a measure of how patterns fill space. Patterns that have more or larger gaps have higher lacunarity. It also quantifies rotational invariance and heterogeneity. A spatial pattern that has a high lacunarity has a high variability of gaps in the pattern, and indicates a more heterogeneous texture [5]. Lacunarity (FD_{lac}) is defined in terms of the ratio of variance over mean value [2].

$$FD_{lac} = \frac{\frac{1}{MN}(\sum_{i=1}^{M-1}\sum_{j=1}^{N-1} P(i, j)^2)}{\{\frac{1}{MN}\sum_{i=1}^{M-1}\sum_{j=1}^{N-1} P(i, j)\}^2} - 1, \tag{4.26}$$

where M and N are the sizes of the fractal dimension image patch P.

4.4.5.3 Diffusing the Dataset

We followed standard engineering practice to select the architecture of our model. To improve the performance of the model, we tried various data augmentation techniques such as label preserving transformation and increasing/decreasing the number minority/majority samples to balance the dataset. We also tried other learning techniques such as one class learning. None of those techniques yielded the desired segmentation results.

Due to discriminative capabilities of fractal dimension and lacunarity features, we used them to diffuse the patch dataset. From experiments, we observed that by diffusing the dataset with these features before normalizing the data yielded a trained model that has better generalization on unseen examples. A comparison of the results obtained with and without dataset diffusion is shown in Fig. 4.6. As can be seen from Table 4.4, when the training dataset was augmented with FD features, there was a huge drop in both error cost during fine-tuning and the classification error during

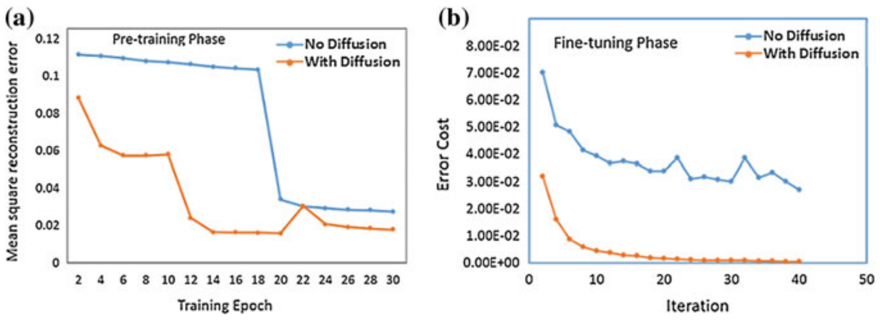


Fig. 4.6 Impact of Data Diffusion on Model Performance. **a** shows that during the pre-training phase, the network achieves lower mean square reconstruction error (msre) when the dataset is diffused with fractal dimension features. Also, as can be seen from **b**, diffusing the dataset leads to faster convergence and lower error cost during the fine-tuning phase

Table 4.4 Data diffusion and network performance

	MSRE	Error cost	Classification error (Training) (%)
Without diffusion	0.0179	7.97e-01	18.51
With diffusion	0.0178	6.0456e-04	0.006

training. It is interesting to note that the reconstruction error almost remained the same in both cases.

4.4.6 *Classification and Segmentation Results*

4.4.6.1 Training, Validation, and Testing

We studied the performance of our model when trained on one latent fingerprint database and tested on another using 3 sets of 20,000 patches, 40% drawn from good, 30% from bad, and 30% from ugly images from NIST, WVU, and IIITD databases. In each of the three experiments, 10,000 patches from a set were used for training, 4,000 for validation, and 6,000 for testing. The results are shown in Table 4.5.

The final training, validation and testing of the model was done with 233,200 patches from the NIST SD27 database with 40% from good, 30% from bad, and 30% from ugly NIST image categories. 132,000 examples were used for training, 48,000 for validation, and 53,200 for testing. Table 4.6 shows the confusion matrix for NIST SD 27 and Table 4.7 shows the TP, TN, FP, and FN, MDR, FDR and classification accuracy on the training, validation, and testing datasets. There was no

Table 4.5 Model performance when trained and tested on different latent fingerprint databases. The numbers in bracket delimited with colon are the training, validation, and testing datasets, respectively. The three datasets are independent. The training and validation datasets shown in column 1 of the last row were obtained exclusively from NIST SD27 database. The testing sets are independent of the training set and were obtained from the target testing database in column 5. MDR_V and FDR_V are the validation MDR and FDR, respectively. Similarly, MDR_T and FDR_T are the testing MDR and FDR, respectively. As shown in the last row, there was a marked improvement in the model performance when more training data was used. When we tried more than 132,000 patches for training, there was no appreciable performance gain despite more training time required to achieve convergence

Train on	Validate on	MDR_V (%)	FDR_V (%)	Test on	MDR_T (%)	FDR_T (%)
NIST SD27 (10,000 : 4,000 : 6,000)	NIST SD27	2.95	1.92	NIST SD27	3.04	1.98
				WVU	3.75	2.25
				IIITD	3.63	2.19
WVU (10,000 : 4,000 : 6,000)	WVU	3.12	2.54	NIST SD27	3.61	3.01
				WVU	3.22	2.87
				IIITD	3.90	3.05
IIITD (10,000 : 4,000 : 6,000)	IIITD	3.32	2.66	NIST SD27	3.49	3.19
				WVU	3.86	3.16
				IIITD	3.28	2.80
NIST SD27 (132,000 : 48,000 : 53,200)	NIST SD27	1.25	0	NIST SD27	1.25	0
				WVU	1.64	0.60
				IIITD	1.35	0.54

Table 4.6 NIST SD27—Confusion matrix for training, validation, and testing

		Predicted patch class (Training)	
		Fingerprint	Non-Fingerprint
Actual patch class	Fingerprint	23,667	9
	Non-Fingerprint	0	108,324
		Predicted patch class (Validation)	
		Fingerprint	Non-Fingerprint
Actual patch class	Fingerprint	12,946	154
	Non-Fingerprint	0	34,900
		Predicted patch class (Testing)	
		Fingerprint	Non-Fingerprint
Actual patch class	Fingerprint	15,291	193
	Non-Fingerprint	0	37,716

Table 4.7 NIST SD27—Training, Validation and Testing Accuracy: Training: **132,000** 8×8 patches; Validation: **48,000** 8×8 patches; Testing: **53,200** 8×8 patches. $MDR = \frac{FN}{TP+FN}$; $FDR = \frac{FP}{TN+FP}$

	TP	TN	FP	FN	MDR (%)	FDR (%)	Classification accuracy (%)
Training	23,667	108,324	0	9	0.04	0	99.96
Validation	12,946	34,900	0	154	1.17	0	98.83
Testing	15,291	37,716	0	193	1.25	0	98.75

noticeable performance gain when the model was trained with more than 132,000 patches.

4.4.6.2 Segmentation Results

Figures 4.7, 4.8, and 4.9 show the segmentation results of our proposed method on sample good, bad, and ugly quality images from the NIST SD27 database. The figures show the original latent fingerprint images and the segmented fingerprints and non-fingerprints constructed using patches classified as fingerprints and non-fingerprints.

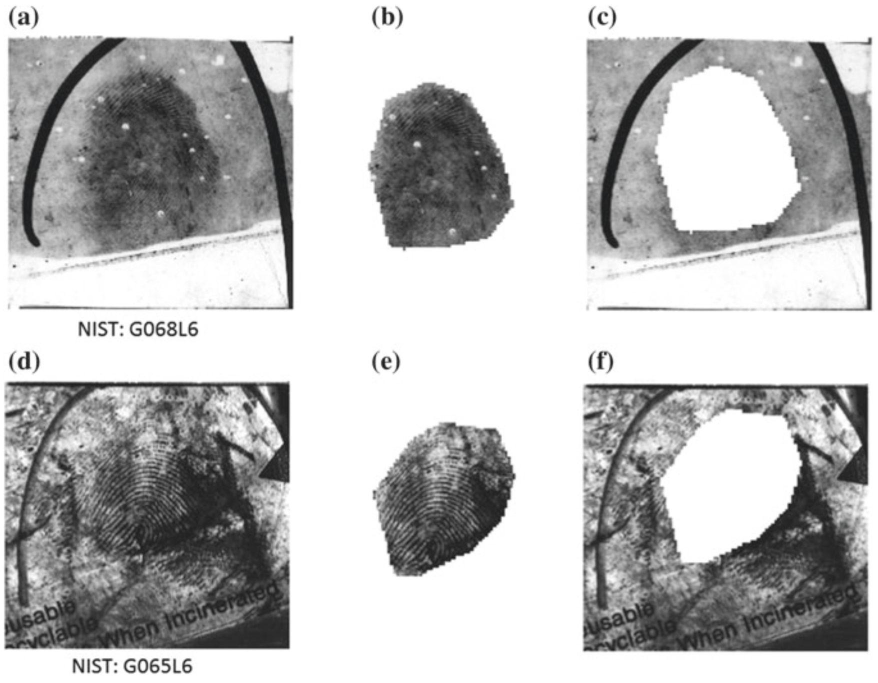


Fig. 4.7 NIST Good Category Latent fingerprint image and segmentation result without post classification processing. **a** and **d** Original images **b** and **e** Fingerprints **c** and **f** Non-fingerprints

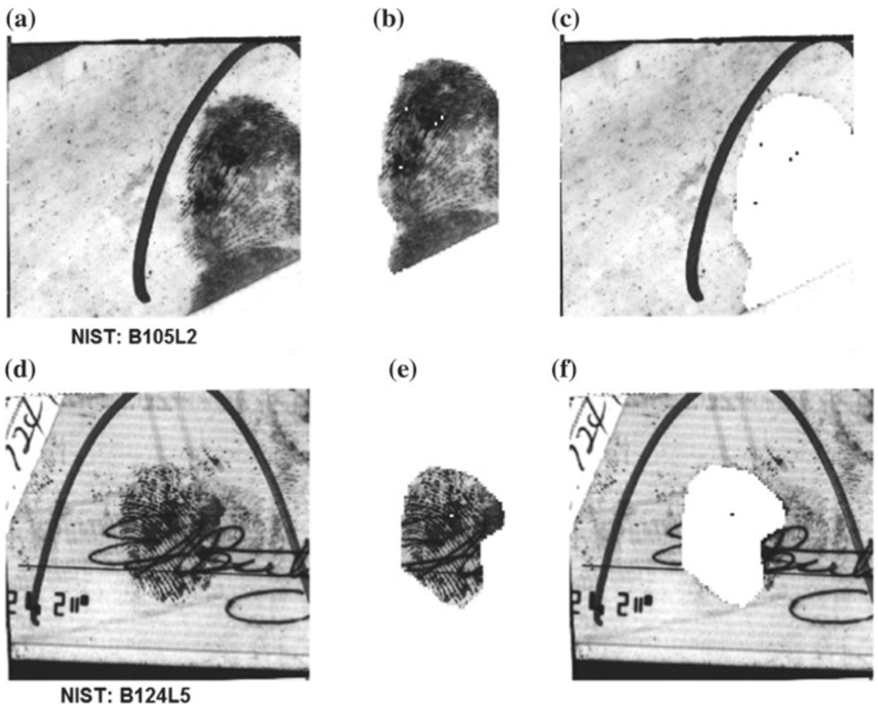


Fig. 4.8 NIST Bad Category [1] Latent Fingerprint Image and segmentation result without post classification processing. **g** and **j** Original images; **h** and **k** Fingerprints **i** and **l** Non-fingerprints

The segmentation results for WVU and IIITD are not shown due to restrictions in the database release agreement (Fig. 4.10).

4.4.7 Comparison with Current Algorithms

Table 4.8 shows the superior performance of our segmentation approach on the good, bad, and ugly quality latent fingerprints from NIST SD27 compared to the results from existing algorithms on the same database. It also shows the performance comparison of our model on WVU and IIITD with other algorithms that reported results on those latent fingerprint databases.

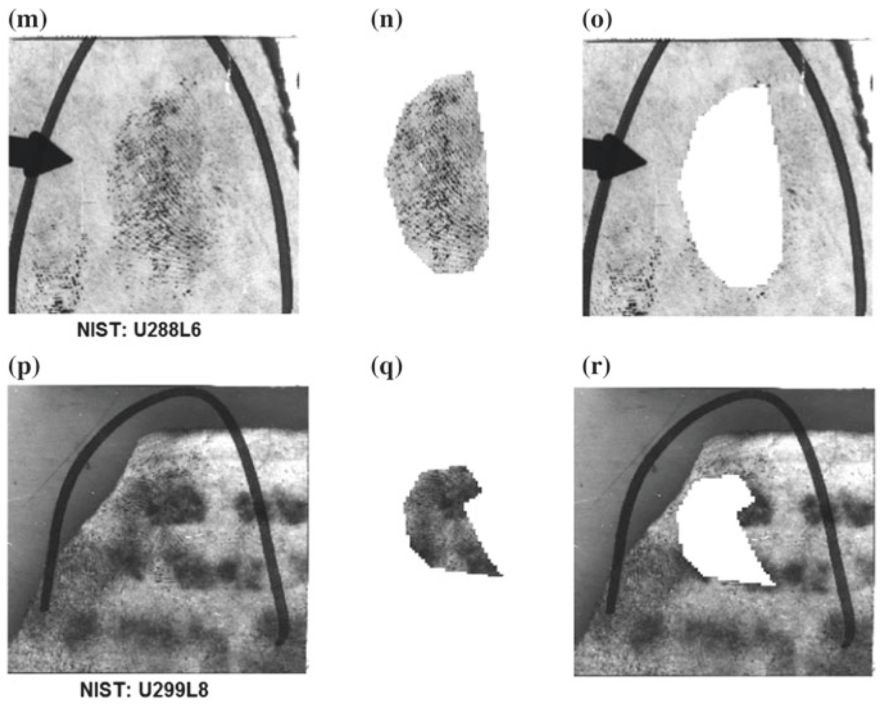


Fig. 4.9 NIST Ugly Category Latent Fingerprint Image and segmentation result without post classification processing. **m** and **p** Original images **n** and **q** Fingerprints **o** and **r** Non-fingerprints

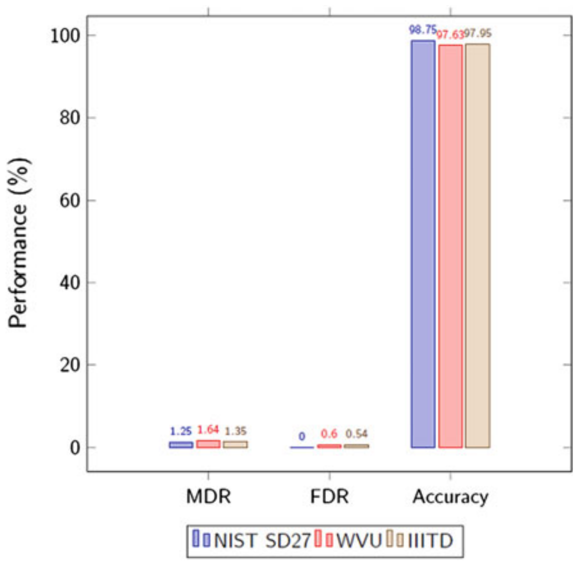


Fig. 4.10 Segmentation reliability in different databases for good quality images. This shows the results of training our model on NIST SD27 and testing on NIST SD27, WVU, and IIITD latent databases. The choice of latent fingerprint database used during training has small impact on the performance of our network. This assertion is also supported by the results in Table 4.5

Table 4.8 Comparison with other algorithms on various datasets

Author	Approach	Database	MDR %	FDR %	Average
Choi et al. [7]	Ridge orientation and frequency computation	NIST SD27	14.78	47.99	31.38
		WVU LDB	40.88	5.63	23.26
Zhang et al. [29]	Adaptive total variation model	NIST SD27	14.10	26.13	20.12
Arshad et al. [4]	K-means clustering	NIST SD27	4.77	26.06	15.42
Jude and Bhanu [9]	Fractal dimension & Weighted ELM	NIST SD27 (Good, Bad, Ugly)	9.22	18.7	13.96
		WVU LDB (Good, Bad, Ugly)	15.54	9.65	12.60
		IIITD LDB (Good)	6.38	10.07	8.23
This chapter	Deep learning	NIST SD27 (Good, Bad, Ugly)	1.25	0.04	0.65
		WVU LDB (Good, Bad, Ugly)	1.64	0.60	1.12
		IIITD (Good)	1.35	0.54	0.95

4.5 Conclusions and Future Work

We proposed a deep architecture based on restricted Boltzmann machine for latent fingerprint segmentation using image patches and demonstrated its performance on the segmentation of latent fingerprint images. The model learns a set of stochastic features that model a distribution over image patches. Using the features extracted from the image patches, the model classifies the patches into fingerprint and non-fingerprint classes. We use the fingerprint patches to reconstruct the latent fingerprint image and discard the non-fingerprint patches which contain the structured noise in the original latent fingerprint. We demonstrated the performance of our model in the segmentation of good, bad, and ugly latent fingerprints from the NIST SD27, as well as WVU and IIITD latent fingerprint databases. We showed that the overall performance of our deep model is superior to that obtained with the state-of-the-art latent fingerprint image segmentation algorithms. Our future work involves developing algorithms for feature extraction and matching for the segmented latent fingerprints.

Acknowledgements This research was supported in part by the Presley Center for Crime and Justice Studies, University of California, Riverside, California, USA.

References

1. NIST Special Database 27. *Fingerprint Minutiae from Latent and Matching Ten-print Images*, http://www.nist.gov/srd/nistsd27.htm
2. O. Al-Kadi, D. Watson, Texture analysis of aggressive and nonaggressive lung tumor CE CT images. IEEE Trans. Biomed. Eng. **55**(7), 1822–1830 (2008)
3. I. Arel, D.C. Rose, T.P. Karnowski, Deep machine learning - a new frontier in artificial intelligence research [research frontier]. IEEE Comput. Intell. Mag. **5**(4), 13–18 (2010)
4. I. Arshad, G. Raja, A. Khan, Latent fingerprints segmentation: feasibility of using clustering-based automated approach. Arabian J. Sci. Eng. **39**(11), 7933–7944 (2014)
5. M. Barros Filho, F. Sobreira, Accuracy of lacunarity algorithms in texture classification of high spatial resolution images from urban areas, in *XXI Congress of International Society of Photogrammetry and Remote Sensing* (2008)
6. M.A. Carreira-Perpinan, G. Hinton, On contrastive divergence learning, in *AISTATS*, vol. 10 (Citeseer, 2005), pp. 33–40
7. H. Choi, A.I.B.M. Boaventura, A. Jain, Automatic segmentation of latent fingerprints, in *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2012), pp. 303–310
8. D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Bengio, Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. **11**, 625–660 (2010)
9. J. Ezeobiejesi, B. Bhanu, Latent fingerprint image segmentation using fractal dimension features and weighted extreme learning machine ensemble, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2016)
10. R.-E. Fan, C.-J. Lin, *A Study on Threshold Selection for Multi-label Classification*. Department of Computer Science (National Taiwan University, 2007), pp. 1–23
11. G. Hinton. A Practical Guide to Training Restricted Boltzmann Machines, Version 1 (2010)
12. C. Huang, C. Moraga, A diffusion-neural-network for learning from small samples. Int. J. Approx. Reason. **35**(2), 137–161 (2004)
13. S. Karimi-Ashtiani, C.-C. Kuo, A robust technique for latent fingerprint image segmentation and enhancement, in *15th IEEE International Conference on Image Processing, 2008, ICIP 2008* (2008), pp. 1492–1495
14. D. Kaye, T. Busey, M. Gische, G. LaPorte, C. Aitken, S. Ballou, L.B. ..., K. Wertheim, Latent print examination and human factors: improving the practice through a systems approach, in *NIST Interagency/Internal Report (NISTIR) - 7842* (2012)
15. J. Keller, S. Chen, R. Crownover, Texture description and segmentation through fractal geometry. Comput. Vis. Graph. Image Process. **45**(2), 150–166 (1989)
16. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
17. M. Lai, Deep learning for medical image segmentation (2015), arXiv:1505.02000
18. A.D.K.T. Lam, Q. Li, Fractal analysis and multifractal spectra for the images, in *2010 International Symposium on Computer Communication Control and Automation (3CA)*, vol. 2 (2010), pp. 530–533
19. P. Lamblin, Y. Bengio, Important gains from supervised fine-tuning of deep architectures on large labeled sets, in *NIPS* 2010 Deep Learning and Unsupervised Feature Learning Workshop* (2010)
20. S. Li, J.T. Kwok, Y. Wang, Combination of images with diverse focuses using the spatial frequency. Inf. Fus. **2**(3), 169–176 (2001)
21. Y. Liu, E. Racah, P.J. Correa, A. Khosrowshahi, D. Lavers, K. Kunkel, M. Wehner, W.D. Collins, Application of deep convolutional neural networks for detecting extreme weather in climate datasets (2016), CoRR abs/1605.01156
22. Z. Makó, Approximation with diffusion-neural-network, in *6th International Symposium of Hungarian Researchers on Computational Intelligence* (2005), pp. 18–19

23. B. Mandelbrot, *The Fractal Geometry of Nature*. Einaudi paperbacks (Henry Holt and Company, New York, 1983)
24. A. Ng, J. Ngiam, C.Y. Foo, http://ufldl.stanford.edu/tutorial/supervised/optimizationstochasticgradientdescent/, *UFLDL Tutorial*
25. A. Sankaran, M. Vatsa, R. Singh, Hierarchical fusion for matching simultaneous latent fingerprint, in *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2012), pp. 377–382
26. N. Sarkar, B.B. Chaudhuri, An efficient differential box-counting approach to compute fractal dimension of image. IEEE Trans. Syst. Man Cybern. **24**(1), 115–120 (1994)
27. N. Short, M. Hsiao, A. Abbott, E. Fox, Latent fingerprint segmentation using ridge template correlation, in *4th International Conference on Imaging for Crime Detection and Prevention 2011 (ICDP 2011)* (2011), pp. 1–6
28. I. Yildirim, Bayesian inference: Gibbs sampling. Technical Note, University of Rochester (2012)
29. J. Zhang, R. Lai, C.-C. Kuo, Latent fingerprint segmentation with adaptive total variation model, in *2012 5th IAPR International Conference on Biometrics (ICB)* (2012), pp. 189–195

Chapter 5
Finger Vein Identification Using Convolutional Neural Network and Supervised Discrete Hashing

Cihui Xie and Ajay Kumar

Abstract Automated personal identification using vascular biometrics, such as from the finger vein images, is highly desirable as it helps to protect the personal privacy and anonymity in during the identification process. The Convolutional Neural Network (CNN) has shown remarkable capability for learning biometric features that can offer robust and accurate matching. We introduce a new approach for the finger vein authentication using the CNN and supervised discrete hashing. We also systematically investigate comparative performance using several popular CNN architectures in other domains, i.e., Light CNN, VGG-16, Siamese and the CNN with Bayesian inference-based matching. The experimental results are presented using a publicly available two-session finger vein images database. Most accurate performance is achieved by incorporating supervised discrete hashing from a CNN trained using the triplet-based loss function. The proposed approach not only achieves outperforming results over other considered CNN architecture available in the literature but also offers significantly reduced template size as compared with those over the other finger vein images matching methods available in the literature to date.

5.1 Introduction

Automated personal identification using unique physiological characteristics of humans, like face, fingerprint, or iris, is widely employed for e-security in a range of applications. In the past decade, there has been significant increase in the detection of surgically altered fingerprints, fake iris stamps, or the usage of sophisticated face masks, to thwart integrity of deployed biometrics systems. Vascular biometrics identification, like using finger vein patterns which are located at about three millimetres below the skin surface, can help to preserve the integrity of biometrics system as it is extremely difficult to surgically alter vascular biometrics. Another advantage of

C. Xie · A. Kumar (✉)
Department of Computing, The Hong Kong Polytechnic University,
Hung Hom, Kowloon, Hong Kong
e-mail: ajay.kumar@polyu.edu.hk

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_5

using finger vein biometrics is related to high degree of personal privacy as finger vein patterns are hidden underneath the skin surface and extremely difficult to acquire then covertly.

The possibility of personal identification using vascular patterns imaged using the light transmitted through hands was indicated in 1992 [6], but was not known to be demonstrated until 2000 [7]. Such earliest work demonstrated feasibility of finger vein identification using normalized-cross correlation. Miura et al. [12] later introduced repeated line tracking approach to improve the performance of finger vein identification, and they further enhanced the performance with maximum curvature [13]. Kumar and Zhou [8] introduced first publicly accessible finger vein database in public domain and comparatively evaluated a range of handcrafted features for the finger vein identification problem. The method introduced in [8] using Gabor filter-based enhancement and morphological operations is still regarded the best performing methods for matching finger vein images. A range of handcrafted features [2, 3, 8–13, 21], primarily obtained from the careful evaluation of the registered images, have been introduced in the literature to investigate finger-vein identification performance. Multiple features acquired from the two cameras [10] or using multiple feature extractors [22] can be combined to significantly improve performance for the vascular matching. One of the limitations of finger vein identification methods introduced in the literature is related to their large template size. Smaller template size is desirable to reduce storage and/or enhance the matching speed for the mobile and online applications. There have also been successful attempts to reduce the finger vein template size, like in [3, 9] or recently in [2] using sparse representation of enhanced finger vein images using the Gabor filters.

The finger vein matching methods available in the literature to date have judiciously introduced handcrafted features and demonstrated promising performance. However, the remarkable capabilities of the deep learning algorithms in automatically learning the most relevant vascular features are yet to be investigated or established. The objective of this work is to fairly investigate the effectiveness of self-learned features using popular convolutional neural network (CNN) architectures and develop more efficient and effective alternative for the automated finger vein identification. The experimental results and comparisons detailed in this chapter used light CNN [20], modified VGG-16 [16], CNN with Bayesian inference, and Siamese network with triplet loss function. Our reproducible [5] experimental results using *publicly* available database indicate that supervised discrete hashing in conjunction with CNN not only achieves outperforming results, but also significantly reduce the finger vein template size which offers increased matching speed. Table 5.1 in the following summarizes promising methods for the finger vein matching that have been introduced in the literature. This table also presents the template size in respective reference, which has been estimated from the details provided in respective reference, performance in terms of EER, and the database used for the performance evaluation. Reference [4] provides good summary of publicly available finger vein image databases introduced in the literature. The usage of two-session databases, particularly for the less constrained or contactless imaging setups as in [8], generates high intra-class variations and is highly desirable to generate fair evaluation of the matching algorithms

Table 5.1 Comparative summary of handcrafted finger vein features in the literature with this work

Ref.	Feature	Database	Two session	Template size (bytes)	EER	No. of subjects	No. of genuine scores	No. of impostor scores
[8]	Handcrafted (Even Gabor)	Public	Yes	106384	4.61%	105	1260	263,340
[8]	Handcrafted (Morphological)	Public	Yes	6710	4.99%	105	1260	263,340
[12]	Handcrafted (Repeated line tracking)	Proprietary	No	43200*	0.145%	678	678*	458,328*
[13]	Handcrafted (Maximum curvature)	Proprietary	No	43200*	0.0009%	678	678*	458,328*
[3]	Handcrafted (Local binary pattern)	Public	No	$\leq$260*	3.53%	156	624	194,064
[2]	Handcrafted (Sparse representation using l1-norm)	Proprietary	No	630*	Unknown	17	Unknown	Unknown
[9]	Handcrafted (Extended local binary pattern)	Public	Yes	131328*	7.22%	105	1260	263,340
[21]	Handcrafted (Unknown algorithm)	Proprietary	Yes	20480*	0.77%	Unknown	10,000	499,500
[11]	Handcrafted (Histogram of salient edge orientation map)	Public	No	$\leq$3496*	0.9%	100	3000	1,797,000
Ours	CNN with triplet similarity loss	Public	Yes	1024	13.16%	105	1260	263,340
Ours	Supervised discrete hashing with CNN	Public	Yes	250	9.77%	105	1260	263,340

*Computed by us from the details available in the respective reference

under more realistic usage/environments. Similarly, the usage of publicly available database can ensure reproducibility of results. Therefore, our all experiments in this chapter incorporate two-session and publicly available database from [8]. The last two rows in this table summarize best performing results from our investigation detailed in this work [19].

The rest of this chapter is organized as follows. Section 5.2 briefly describes on the preprocessing of the finger vein images and includes relevant steps for the image normalization, segmentation and enhancement. The CNN architectures, LCNN, VGG, LCNN with triplet similarity loss function, and LCNN with joint Bayesian formulation investigated in this are introduced in Sect. 5.3 while Sect. 5.4 details the supervised discrete hashing algorithm investigated to enhance performance and reduce the template size. The experimental results are presented in Sect. 5.4 and includes discussion on our findings, comparison with earlier methods. Finally, the key conclusions from this work are summarized in Sect. 5.5.

5.2 Image Normalization for Finger Vein Images

5.2.1 Preprocessing

Acquisition of finger vein images can introduce translational and rotational changes in different images acquired from the same finger or subject. Therefore, automated extraction of fixed region of interest (ROI) that can minimize such intra-class variations is highly desirable. The method of ROI localization considered in this work is same as detailed in [8] as it works well in most cases. Figure 5.1 illustrates samples of the acquired images using the near infrared camera.

Once the region of interest is localized, we can recover the binary masks corresponding to the ROI which can be used for alignment of finger vein samples, so that the adverse influence from the rotational changes in fingers can be minimized. The method for estimating the rotation is same as described in [8]. This estimated angle is used to align ROI, before the segmentation, in a preferred direction.

5.2.2 Image Enhancement

The finger vein details from the normalized images are subjected to the contrast enhancement to enhance clarity in vascular patterns which can be more reliable for

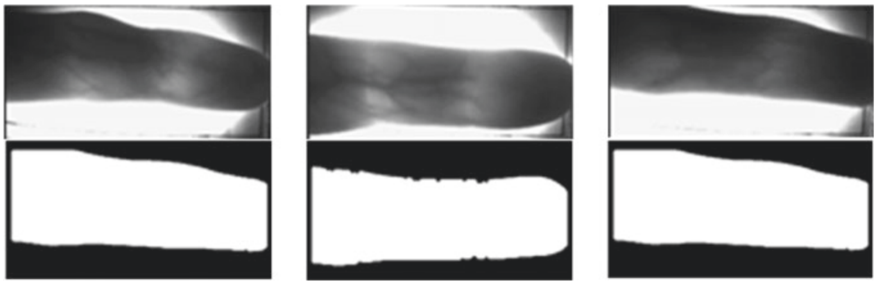


Fig. 5.1 Finger vein image samples before preprocessing (*first row*) and the binary images generated during the preprocessing (*second row*) stage

the training. Since vascular patterns are generally continuous, if a pixel belongs to the vascular pattern, there is a high possibility that its surrounding pixels are also part of the same vascular pattern and have similar gray level. Such observation is the same for nonvascular parts. Therefore, enhancement by computing the average gray level surrounding a pixel can help to enlarge the difference between the vascular parts and nonvascular parts, and makes the finger vein details more obvious as a result. After such enhancement, the vascular patterns become clearer with details as shown from sample images in Fig. 5.2.

The vascular patterns in the normalized image samples can be further enhanced by spatial filtering from orientation selective band pass filters, similar to as used in the enhancement of fingerprint images. We also attempted to ascertain usefulness of such enhanced finger vein images using the Gabor filters. These filters from the twelve different orientations are selected to generate enhanced finger vein images as shown in Fig. 5.4. Such enhanced images [8] using Gabor filters are effective in accentuating the vascular features and therefore its possible usage in automatically vascular features (Fig. 5.3) from CNN was also investigated in the experiments.

The finger vein image-processing operations essentially generates two kinds of enhanced images, ROI-A and ROI-B shown in Fig. 5.4, that were considered for the performance evaluation using the CNNs.

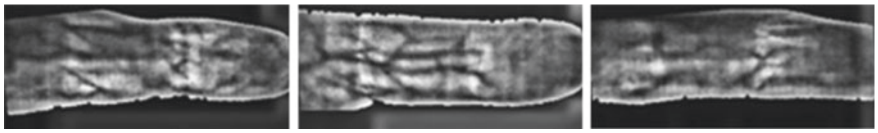


Fig. 5.2 Enhanced ROI vein images after rotation

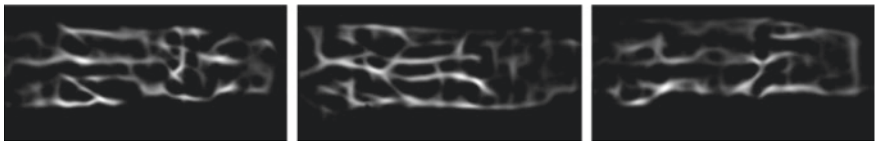


Fig. 5.3 Samples from even Gabor filtered finger vein images

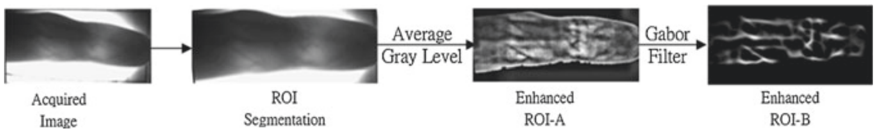


Fig. 5.4 Key steps in the generation of enhanced finger vein images for the CNNs

5.3 Convolutional Neural Network Architectures

A variety of models for the deep learning have been developed to learn useful feature representation but largely for the face biometric image patterns. A variety of such models using CNN have been introduced in the literature and were investigated to ascertain performance for the finger vein image matching. A brief introduction to various CNN architectures considered in this work is provided in the following sections.

5.3.1 Light CNN

The light CNN (LCNN) framework introduces a Max-Feature-Map (MFM) operation [20] between convolutional layers which establish a competitive relationship for superior generalization capability and reduce parameter space (compact feature representation). Such *maxout* activation (Fig. 5.5) function significantly reduces complexity and makes CNN lighter, where *conv* stands for convolutional layer.

For a convolutional layer without MFM, suppose that input size is $N_1 \times W_1 \times H_1$ and output size is $N_2 \times W_2 \times H_2$ then the required complexity using big 'O' notation can be represented as follows:

$$O\left(N_1 N_2 \Omega\right) \;\; where\ \Omega = W_1 \times H_1 \times W_2 \times H_2 \tag{5.1}$$

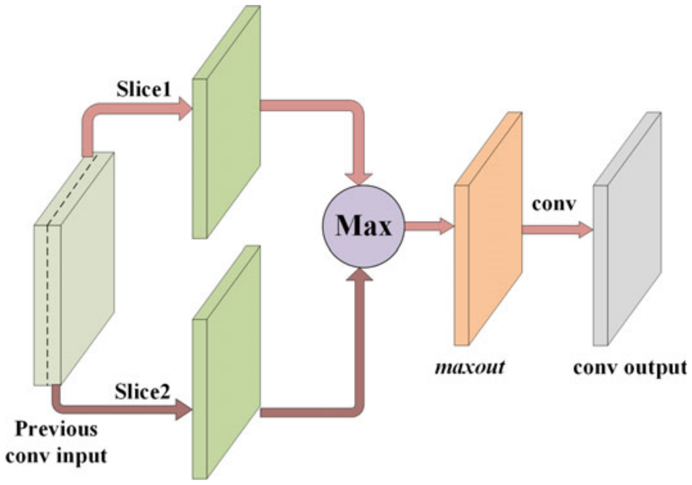


Fig. 5.5 Illustration for computing the Max-Feature Map in LCNN

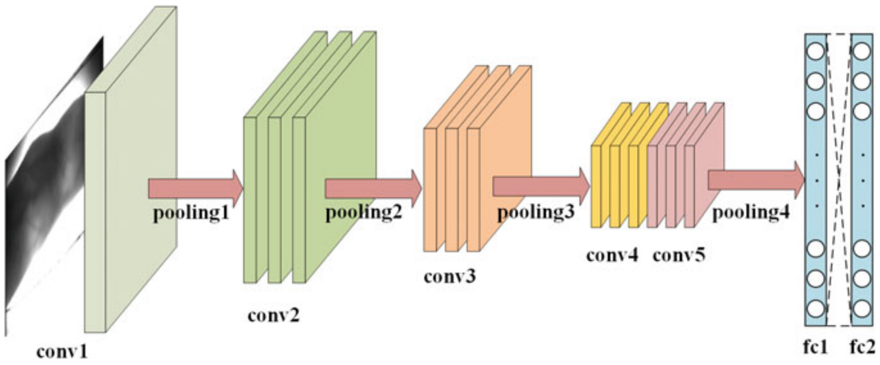


Fig. 5.6 The architecture for LCNN investigated in our experiments

For a convolutional layer with MFM, we could slice input into two equal-size parts, each with $\frac{N_1}{2} \times W_1 \times H_1$. Then for each corresponding element in two sliced parts we generate an output using *maxout* activation, which has a size of $\frac{N_1}{2} \times W_1 \times H_1$. With this smaller or lighter data as the input, complexity of convolutional layer reduces to

$$O\left(\frac{N_1 N_2 \Omega}{2}\right) \ where\ \Omega = W_1 \times H_1 \times W_2 \times H_2 \tag{5.2}$$

Comparing (5.1) and (5.2) we can infer that the usage of MFM can help to significantly reduce the complexity or make CNN lighter.

The loss function we used in this structure is *softmax* loss function. The basic idea is to combine *softmax* function with a negative log-likelihood, and the last layer information is used to estimate the identity of the class.

The architecture of LCNN employed in our experiments is shown in Fig. 5.6 (MFM part is excluded to maintain the clarity). This network contains 9 convolutional layers (conv), 4 pooling layers (pooling) and 2 fully connected layers (fc) and some assistant layers which are summarized in Table 5.2.

5.3.2 LCNN with Triplet Similarity Loss Function

Deep Siamese networks have been successfully incorporated to learn a similarity metric between a pair of images. We incorporated similar triplet similarity loss function as detailed in [14] for LCNN to learn the similarity metric.

We randomly select an image x^r from training set as *random* sample in Fig. 5.7. Then, we choose image x^p which is from the same class referred to as *positive* sample and image x^n which is from a different class referred to as *negative* sample. After LCNN, we get the features $f(x^r)$, $f(x^n)$, and $f(x^p)$. Our objective is to decrease the

Table 5.2 Details of layer information of LCNN

Index	Type	Filter size	Num	Stride	Pad
1	conv1	5	96	1	2
2	MFM1	–	48	–	–
3	pooling1	2	–	2	0
4	conv2a	1	96	1	0
5	MFM2a	–	48	–	–
6	conv2	3	192	1	1
7	MFM2	–	96	–	–
8	pooling2	2	–	2	0
9	conv3a	1	192	1	1
10	MFM3a	–	96	–	–
11	conv3	3	384	1	1
12	MFM3	–	192	–	–
13	pooling3	2	–	2	0
14	conv4a	1	384	1	1
15	MFM4a	–	192	–	–
16	conv4	3	256	1	1
17	MFM4	–	128	–	–
18	conv5a	1	256	1	1
19	MFM5a	–	128	–	–
20	conv5	3	256	1	1
21	MFM5	–	128	–	–
22	pooling4	2	–	2	0
23	fc1	–	512	–	–
24	MFMfc	–	256	–	–
25	fc2	–	500	–	–

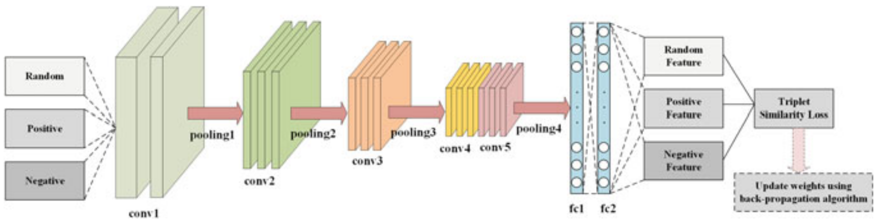


Fig. 5.7 The architecture for LCNN with triplet similarity loss function

similarity distance between *random* and *positive* features, and increase it between *random* and *negative* features, which indicates why it is named as triplet similarity loss. At the same time, we also need to ensure that there is a sufficient *margin* between them.

Suppose we have a random set $\mathbf{X^r} = \{x_i^r\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ and its corresponding positive set $\mathbf{X^p} = \{x_i^p\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ and negative set $\mathbf{X^n} = \{x_i^n\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$. Considering these notations, we can write our loss function as follows:

$$\sum\nolimits_{i=1}^{N} [\| f\left(x_i^r\right) - f\left(x_i^p\right) \|^2 - \| f\left(x_i^r\right) - f\left(x_i^n\right) \|^2 + margin]_+ \tag{5.3}$$

where $[\cdot]_+$ presents that we maintain positive values and change others to zero. The architecture of LCNN with such triplet similarity loss function is shown in Fig. 5.7 and detailed in Table 5.3. When the set of input consists of n random samples, n positives and n negatives, we generate $3n \times 500$ features. These pairs were split into three parts, each with the size of $n \times 500$, and used as the input for computing triplet similarity loss for updating the neuron weights during the network training.

5.3.3 Modified VGG-16

The Visual Geometry Group architecture with 16 layers (VGG-16) [16] was modified for the CNN to directly recover the match scores, instead of the feature vectors, in our experiment. Our modification was motivated to fit the rectangular finger vein ROI images without introducing the distortion. We used pair of images rather than single image as input in conventional VGG-16 since we want to compare the similarity between two finger vein images. The input image size is also different from conventional VGG-16, which is 224 × 224, while it's 128 × 488 pixels for our finger vein ROI images. The training phase utilized the cross-entropy loss function which can be written as follows:

$$-\frac{1}{n}\sum\nolimits_{i=1}^{n}\left[y_i \log\left(\widehat{\boldsymbol{y}}_{\boldsymbol{l}}\right) + (1 - y_i)\log\left(1 - \widehat{\boldsymbol{y}}_{\boldsymbol{l}}\right)\right] \tag{5.4}$$

where $\widehat{\boldsymbol{y}}_{\boldsymbol{l}} = g(\boldsymbol{w}^{\boldsymbol{T}}\boldsymbol{x}_{\boldsymbol{i}})g(\cdot)$ is the logistic function, $\boldsymbol{x}_{\boldsymbol{i}}$ is the extracted feature and $\mathbf{w}$ is the weight that needs optimized during the training. The architecture of Modified VGG-16 (MVGG) is shown in Fig. 5.8 and Table 5.4.

5.3.4 LCNN with Joint Bayesian Formulation

The principal component analysis (PCA) is a classical method to extract the most important features and is popular for the dimensionality reduction of the features. In another set of experiments, we incorporated PCA for the dimensionality reduction of features extracted from LCNN and then employed joint Bayesian [1] approach as distance metrics for matching finger vein images.

For any feature $\boldsymbol{f}$ extracted from LCNN, we regard it as combination of two parts $\boldsymbol{\mu}$ and $\boldsymbol{\varepsilon}$ where $\boldsymbol{\mu}$ is the average feature of the class to which $\boldsymbol{f}$ belongs and $\boldsymbol{\varepsilon}$ is

Table 5.3 Details of layer information of LCNN with triplet similarity loss

Index	Type	Input index	Output index	Filter size	Output size	Stride	Pad
1	DataR	–	4	–	N*W*H	–	–
2	DataP	–	4	–	N*W*H	–	–
3	DataN	–	4	–	N*W*H	–	–
4	Data	1,2,3	5	–	3N*W*H	–	–
5	conv1	4	6	5	3N*96	1	2
6	MFM1	5	7	–	3N*48	–	–
7	pooling1	6	8	2	–	2	0
8	conv2a	7	9	1	3N*96	1	0
9	MFM2a	8	10	–	3N*48	–	–
10	conv2	9	11	3	3N*192	1	1
11	MFM2	10	12	–	3N*96	–	–
12	pooling2	11	13	2	–	2	0
13	conv3a	12	14	1	3N*192	1	1
14	MFM3a	13	15	–	3N*96	–	–
15	conv3	14	16	3	3N*384	1	1
16	MFM3	15	17	–	3N*192	–	–
17	pooling3	16	18	2	–	2	0
18	conv4a	17	19	1	3N*384	1	1
19	MFM4a	18	20	–	3N*192	–	–
20	conv4	19	21	3	3N*256	1	1
21	MFM4	20	22	–	3N*128	–	–
22	conv5a	21	23	1	3N*256	1	1
23	MFM5a	22	24	–	3N*128	–	–
24	conv5	23	25	3	3N*256	1	1
25	MFM5	24	26	–	3N*128	–	–
26	pooling4	25	27	2	–	2	0
27	fc1	26	28	–	3N*512	–	–
28	MFMfc	27	29	–	3N*256	–	–
29	fc2	28	30,31,32	–	3N*500	–	–
30	SliceR	29	33	–	N*500	–	–
31	SliceP	29	33	–	N*500	–	–
32	SliceN	29	33	–	N*500	–	–
33	Loss	30,31,32	–	–	–	–	–

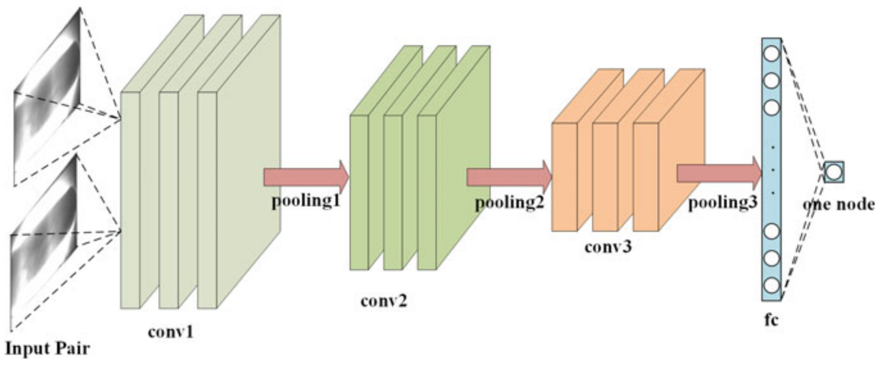


Fig. 5.8 The architecture for Modified VGG-16 for finger vein image matching

Table 5.4 Details of layer information of Modified VGG-16

Index	Type	Filter size	Num	Stride	Pad
1	conv1a	3	64	1	1
2	ReLU1a	–	–	–	–
3	conv1b	3	64	1	1
4	ReLU1b	–	–	–	–
5	conv1c	3	64	1	1
6	ReLU1c	–	–	–	–
7	pooling1	2	–	2	0
8	conv2a	3	128	1	1
9	ReLU2a	–	–	–	–
10	conv2b	3	128	1	1
11	ReLU2b	–	–	–	–
12	conv2c	3	128	1	1
13	ReLU2c	–	–	–	–
14	pooling2	2	–	2	0
15	conv3a	3	256	1	1
16	ReLU3a	–	–	–	–
17	conv3b	3	256	1	1
18	ReLU3b	–	–	–	–
19	conv3c	3	256	1	1
20	ReLU3c	–	–	–	–
21	pooling3	2	–	2	0
22	fc1	–	512	–	–
23	Dropout	–	–	–	–
24	fc2	–	1	–	–

the intra-class variation, and we suppose that $\boldsymbol{\mu}$ and $\boldsymbol{\varepsilon}$ are two independent variables with Gaussian distribution $N(0, \boldsymbol{\sigma}_{\boldsymbol{\mu}})$ and $N(0, \boldsymbol{\sigma}_{\boldsymbol{\varepsilon}})$.

Let I be the hypothesis that $\boldsymbol{f}_1$ and $\boldsymbol{f}_2$ are from the same class, and E mean that they are from different class. Thus we could write the goal as to enlarge $\frac{P(f_1 f_2|I)}{P(f_1 f_2|E)}$ where $P(\cdot)$ is the distribution. For simplicity, we use log formulation $r(\boldsymbol{f}_1, \boldsymbol{f}_2) = \log\frac{P(f_1 f_2|I)}{P(f_1 f_2|E)}$ and the computations are as detailed in [1].

Our objective has been to enlarge $r(\boldsymbol{f}_1, \boldsymbol{f}_2)$ and therefore we compute maximum of results rather than the minimum while using the L2-norm. The CNN training part is the same as LCNN. After extraction of features, we retain 80 dimensions instead of original 500 to accelerate computations and use these features to compute matrices for the joint Bayesian formulation based classification.

5.4 Supervised Discrete Hashing

One of the key challenges for the successful usage of biometrics technologies are related to efficient search speed (fast retrieval) and template storage/size. Hashing is one of the most effective approaches to address such challenges and can *efficiently* encode the biometrics templates using binary numbers (2000 in our experiments) that closely reflect similarity with the input data/templates. With such strategy we can only store the corresponding short/compact binary codes, instead of original feature templates, and significantly improve the search or the matching speed by highly efficient pairwise comparisons using the Hamming distance.

This framework for an effective supervised hashing scheme is introduced in [15] and the objective in the learning phase is to generate binary codes for the linear classification. We firstly define the problem and assume that we have n samples/features $\mathbf{X} = [\mathbf{x}_1\mathbf{x}_2 \ldots \mathbf{x}_n]$ and our goal is to recover corresponding binary codes $\mathbf{B} = [\mathbf{b}_1\mathbf{b}_2 \ldots \mathbf{b}_n]$ where $\mathbf{b}_i \in \{-1, 1\}$, $i = 1, 2, \ldots, n$. Since we have labels, in order to make good use of these information, we define a multi-class classification function:

$$\hat{\mathbf{y}} = \mathbf{W}^{\mathrm{T}}\mathbf{b} \text{ where } \mathbf{W} = [\mathbf{w}_1\mathbf{w}_2 \ldots \mathbf{w}_C], \tag{5.5}$$

where C is the total number of classes, and $\hat{\mathbf{y}} \in \mathbb{R}^{C\times 1}$ is the label vector, where the maximum one indicates its class of input $\mathbf{x}$. Now we can formulate the hashing problem as follows:

$$\min_{\boldsymbol{B}, \boldsymbol{W}, F} \sum_{i=1}^{n} L(\boldsymbol{y}_i, \mathbf{W}^{\mathbf{T}}\mathbf{b}_i) + \lambda\|\mathbf{W}\|^2, \text{ s.t. } \mathbf{b}_i = \operatorname{sgn}(F(\mathbf{x}_i)) \tag{5.6}$$

where $L(\cdot)$ represents the loss function used by us which is the L2-norm in our experiments, λ is the regularization parameter, and at the same time, $\mathbf{b}_i$ is generated by the hash function $\operatorname{sgn}(F(\mathbf{x}_i))$ where $\operatorname{sgn}(\cdot)$ is the sign function. With the help of Lagrange Multiplier, we can then rewrite (5.6) as:

$$\min_{\boldsymbol{B},\boldsymbol{W},F} \sum_{i=1}^{n} L(\boldsymbol{y}_i, \mathbf{W}^{\mathbf{T}}\mathbf{b}_i) + \lambda\|\mathbf{W}\|^2 + \mu \sum_{i=1}^{n} \|\mathbf{b}_i - F(\mathbf{x}_i)\|^2 \tag{5.7}$$

where μ is the Lagrange multiplier. We further select a nonlinear form of function for $F(\mathbf{x})$:

$$F(\mathbf{x} = \mathbf{U}^{\mathrm{T}}\phi(\mathbf{x}) \tag{5.8}$$

where $\mathbf{U}$ is the parameter matrix and $\phi(\mathbf{x})$ is a k-dimensional kernel that

$$\phi(\mathbf{x}) = \begin{bmatrix} \exp(\|\frac{\mathbf{x}-a_1\|^2}{\sigma}) \\ \cdots \\ \exp(\|\frac{\mathbf{x}-a_k\|^2}{\sigma}) \end{bmatrix} \tag{5.9}$$

$\boldsymbol{a}_j, j = 1, 2, \ldots, k$ are randomly selected anchor vectors from input. In order to compute $\boldsymbol{U}$ in the function, we can rewrite (5.6) as follows:

$$\min_{\mathbf{U}} \sum_{i=1}^{n} \|\mathbf{b}_i - F(\mathbf{x}_i)\|^2 = \min_{\mathbf{U}} \|\mathbf{U}^{\mathrm{T}}\Phi(\mathbf{X}) - \mathbf{B}\|^2 \tag{5.10}$$

where $\Phi(\mathbf{X}) = \{\phi(\mathbf{x}_i)\}_{i=1}^{n}$ and our purpose is to set the gradient to zero, which is

$$\nabla_{\mathbf{U}}\left(\|\mathbf{U}^{\mathrm{T}}\Phi(\mathbf{X}) - \mathbf{B}\|^2\right) = 2\left(\mathbf{U}^{\mathrm{T}}\Phi(\mathbf{X}) - \mathbf{B}\right)\Phi(\mathbf{X})^{\mathrm{T}} = 0 \tag{5.11}$$

It is simpler to achieve the final computation for $\mathbf{U}$ as follows:

$$\mathbf{U} = (\Phi(\mathbf{X})\,\Phi(\mathbf{X})^{\mathrm{T}})^{-1}\Phi(\mathbf{X})\,\mathbf{B}^{\mathrm{T}} \tag{5.12}$$

In order to solve for $\mathbf{W}$, we make use of the same method, first simplify (5.6) to

$$\min_{\mathbf{W}} \sum_{i=1}^{n} L(\boldsymbol{y}_i, \mathbf{W}^{\mathbf{T}}\mathbf{b}_i) + \lambda\|\mathbf{W}\|^2 = \min_{\mathbf{W}} \|\mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 + \lambda\|\mathbf{W}\|^2, \tag{5.13}$$

and then calculate its gradient based on $\mathbf{W}$

$$\nabla_{\mathbf{W}}(\|\mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 + \lambda\|\mathbf{W}\|^2) = 2\mathbf{B}\left(\mathbf{B}^{\mathrm{T}}\mathbf{W} - \mathbf{Y}^{\mathrm{T}}\right) + 2\lambda\mathbf{W}, \tag{5.14}$$

which can be set as zero and we get

$$\mathbf{W} = \left(\mathbf{B}\mathbf{B}^{\mathrm{T}} + \lambda\mathbf{I}\right)^{-1}\mathbf{B}\mathbf{Y}^{\mathrm{T}}, \tag{5.15}$$

Finally, we can solve for $\mathbf{B}$ and we exclude those variables which have no relation to $\mathbf{B}$ and then rewrite (5.6) as follows.

$$\min_{\mathbf{B}} \|\mathbf{Y} - \mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 + \mu\|\mathbf{B} - F(\mathbf{X})\|^2 = \min_{\mathbf{B}} \|\mathbf{Y}\|^2 - 2\mathrm{tr}\left(\mathbf{Y}^{\mathrm{T}}\mathbf{W}^{\mathrm{T}}\mathbf{B}\right)$$
$$+ \|\mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 + \mu(\|\mathbf{B}\|^2 + 2\mathrm{tr}\left(\mathbf{B}^{\mathrm{T}}F(\mathbf{X})\right) + \|F(\mathbf{X})\|^2) \tag{5.16}$$

or we can further simplify the above formulation as follows:

$$\min_{\mathbf{B}} \|\mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 - 2\mathrm{tr}\left(\mathbf{B}^{\mathrm{T}}(F(\mathbf{X}) + \mathbf{W}\mathbf{Y})\right) \tag{5.17}$$

$\|\mathbf{B}\|^2$ is excluded here because $\mathbf{b}_i \in \{-1, 1\}, i = 1, 2, \ldots, n$, indicating that $\|\mathbf{B}\|^2$ is some constant.

We can now solve this problem *bit-by-bit*. Let $\mathbf{p}^{\mathrm{T}}$ represent the lth row of $\mathbf{B}$, and $\mathbf{B}'$ is the matrix without $\mathbf{p}^{\mathrm{T}}$. Similarly let $\mathbf{v}^{\mathrm{T}}$ be the lth row of $\mathbf{W}$ and let $\mathbf{q}^{\mathrm{T}}$ be the lth row of $\mathbf{Q}$ where $\mathbf{Q} = F(\mathbf{X}) + \mathbf{W}\mathbf{Y}$ then we can ignore $\mathbf{W}'$ and $\mathbf{Q}'$ While moving a row to the end for all matrices would not cause problems but help to better understand the problem. In order to enhance clarity of the problem, we can move all the lth row to the end and rewrite $\mathbf{B} = \begin{bmatrix} \mathbf{B}' \\ \mathbf{p}^{\mathrm{T}} \end{bmatrix}$, and the same for $\mathbf{W}$ and $\mathbf{Q}$ We can then rewrite first term in (5.17) as follows.

$$\|\mathbf{W}^{\mathrm{T}}\mathbf{B}\|^2 = \left\|\begin{bmatrix} \mathbf{W}'^{\mathrm{T}} & \mathbf{v} \end{bmatrix}\begin{bmatrix} \mathbf{B}' \\ \mathbf{p}^{\mathrm{T}} \end{bmatrix}\right\|^2 = \|\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\|^2 + \|\mathbf{v}\mathbf{p}^{\mathrm{T}}\|^2 + 2\mathrm{tr}\left(\mathbf{B}'^{\mathrm{T}}\mathbf{W}'\mathbf{v}\mathbf{p}^{\mathrm{T}}\right)$$
$$= \|\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\|^2 + \mathrm{tr}\left(\mathbf{v}\mathbf{p}^{\mathrm{T}}\mathbf{p}\mathbf{v}^{\mathrm{T}}\right) + 2\left(\mathbf{W}'\mathbf{v}\right)^{\mathrm{T}}\mathbf{B}'\mathbf{p} \tag{5.18}$$

While $\|\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\|^2 + \mathrm{tr}\left(\mathbf{v}\mathbf{p}^{\mathrm{T}}\mathbf{p}\mathbf{v}^{\mathrm{T}}\right)$ is equal to some constant, and because our goal is to solve for $\mathbf{p}$ we can regard other parts as constant. Here $\|\mathbf{v}\mathbf{p}^{\mathrm{T}}\|^2$ is ignored because $\|\mathbf{v}\mathbf{p}^{\mathrm{T}}\|^2 = \|\mathbf{p}\mathbf{v}^{\mathrm{T}}\|^2 = \mathrm{tr}\left(\mathbf{v}\mathbf{p}^{\mathrm{T}}\mathbf{p}\mathbf{v}^{\mathrm{T}}\right) = n\,\mathrm{tr}\left(\mathbf{v}\mathbf{v}^{\mathrm{T}}\right)$ where $\mathbf{p}^{\mathrm{T}}\mathbf{p} = n$. The other part can be simplified as follows:

$$\mathrm{tr}\left(\mathbf{B}^{\mathbf{T}}\mathbf{Q}\right) = \mathrm{tr}\left(\mathbf{B}'^{\mathbf{T}}\mathbf{Q}' + \mathbf{p}\mathbf{q}^{\mathrm{T}}\right) = \mathrm{tr}\left(\mathbf{B}'^{\mathbf{T}}\mathbf{Q}') + \mathrm{tr}(\mathbf{p}\mathbf{q}^{\mathrm{T}}\right) = \mathrm{tr}(\mathbf{B}'^{\mathbf{T}}\mathbf{Q}') + \mathbf{q}^{T}\boldsymbol{p} \tag{5.19}$$

Combining these terms, we can rewrite (5.17) as follows.

$$\min_{\mathbf{B}} \mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}'\mathbf{p} - \mathbf{q}^{\mathrm{T}}\mathbf{p} = \min_{\mathbf{B}} (\mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}' - \mathbf{q}^{\mathrm{T}})\mathbf{p} \tag{5.20}$$

This is an optimization problem, and $\mathbf{p} \in \{-1, 1\}^n$, therefore we just need to incorporate the opposite sign for its first argument.

$$\mathbf{p} = -\mathrm{sgn}\left(\mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}' - \mathbf{q}^{\mathrm{T}}\right) = \mathrm{sgn}(\mathbf{q}^{\mathrm{T}} - \mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}') \tag{5.21}$$

We can now *explicitly* outline computed parts in the following.

$$\mathbf{W} = \left(\mathbf{B}\mathbf{B}^{\mathrm{T}} + \lambda\mathbf{I}\right)^{-1}\mathbf{B}\mathbf{Y}^{\mathrm{T}}, \tag{5.15}$$

$$\mathbf{U} = (\Phi\left(\mathbf{X}\right)\Phi\left(\mathbf{X}\right)^{\mathrm{T}})^{-1}\Phi\left(\mathbf{X}\right)\mathbf{B}^{\mathrm{T}} \tag{5.12}$$

$$\mathbf{p} = \mathrm{sgn}(\mathbf{q}^{\mathrm{T}} - \mathbf{v}^{\mathrm{T}}\mathbf{W}'^{\mathrm{T}}\mathbf{B}') \tag{5.21}$$

Shen et al. [15] have provided another computation based on the hinge loss. However for the simplicity, we incorporated L2-norm in our experiments and therefore this part is excluded here. We now have the required equations here and can *summarize* this algorithm as follows.

```
Algorithm: Supervised Discrete Hashing

Input: Training data {$\mathbf{X}$, $\mathbf{Y}$}
Output: Binary codes $\mathbf{B}$

1. Randomly select k anchors $\boldsymbol{a}_j$, $j = 1, 2, \ldots, k$ from $\mathbf{X}$ and calculate $\Phi(\mathbf{X})$
2. Randomly initiate $\mathbf{B}$
3. Loop until converge or reach maximum iterations
    – Calculate $\mathbf{W}$ and $\mathbf{U}$ which are described in (5.15) and (5.12)
    – Learn $\mathbf{B}$ bit by bit, with the help of (5.21)
```

5.5 Experiments and Results

This section provides details on the experiments performed using various CNN architectures discussed in previous sections.

5.5.1 Database and Evaluation Protocol

In order to ensure reproducibility of experimental results, we utilized publicly available *two-session* finger vein images database from [17]. This database of 6264 images has been acquired from 156 different subjects and includes finger vein images from two fingers for each subject. However, second session images are only from 105 different subjects. This database has high intra-class variations in the images and includes significant variations in the quality of images which makes it most suitable for benchmarking the finger vein matching algorithms for the real applications. In our experiments, we only used first session images to train different network architectures discussed in previous section, and initially excluded 51 subjects without second session images. The experimental results are presented using independent second session test data. Therefore, each of the receiver operating characteristics uses 1260 (210×6) genuine scores and 263340 ($210 \times 209 \times 6$) impostor scores.

We experimented on ROI images, enhanced ROI images, and even Gabor- filtered images separately. The ROI images have 256×513 pixels, enhanced ROI images have 218×488 pixels, and even Gabor filtered images have 218×488 pixels. The experimental results using ROC and CMC from the respective CNN architecture are presented in the following.

5.5.2 Results Using LCNN

We first performed the experiments using the LCNN trained using the ROI images, enhanced ROI images, and the enhanced images using even Gabor filters (Figs. 5.2, 5.3, and 5.4). The experimental results using respective second session dataset are shown in Fig. 5.9. The respective ROC and CMC illustrate that *enhanced* ROI images can achieve superior matching performance than those from using ROI images. The enhanced ROI images using even Gabor filters significantly helps to *suppress* the noisy pixels and accentuate the vascular regions and is the plausible reason for superior accuracy.

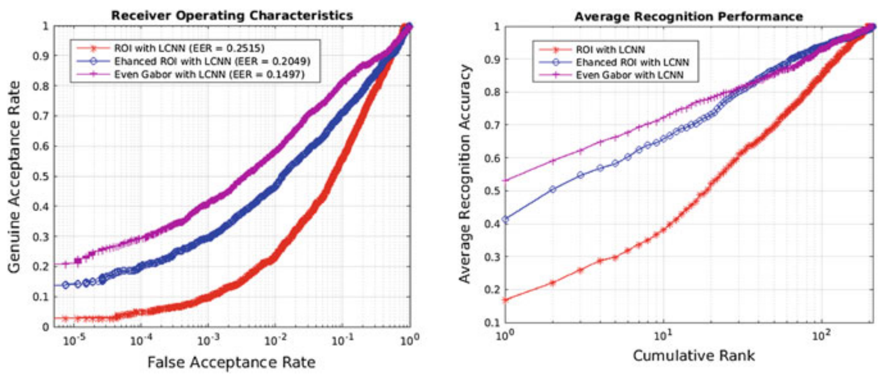


Fig. 5.9 Comparative ROC (*left*) and CMC (*right*) performance using LCNN

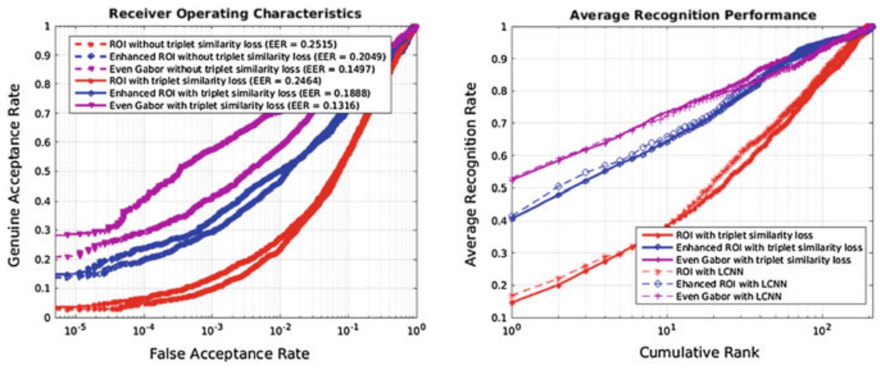


Fig. 5.10 Comparative ROC (*left*) and CMC (*right*) performance using LCNN with triplet similarity loss

5.5.3 Results Using LCNN with Triplet Similarity Loss Function

The experimental results using LCNN trained with Siamese triplet similarity loss function are presented in Fig. 5.10. These results consistently illustrate superior performance using the architecture than the LCNN. The performance from the ROC of enhanced ROI with Gabor filters is *significantly* superior and this observation is in line with the trend observed from results using LCNN in Fig. 5.9 (the dash lines in Fig. 5.10 are previous results using LCNN for ease in the comparison). However, this approach has little influence on CMC.

The LCNN *without* triplet similarity loss tries to match a sample with its label, while LCNN *with* similarity loss focuses on the similarities of the images which could contribute to the better ROC performance. However, at the same time, label information is not sufficiently exploited with the triplet similarity loss and therefore the CMC performance has not changed significantly.

5.5.4 Results Using CNN and Joint Bayesian Formulation

Another scheme that has shown superior performance for ocular classification in [23] uses joint Bayesian [1] instead of L2-norm as the metrics for the similarity. The LCNN with the joint Bayesian classification scheme was also attempted to ascertain the performance. The ROC using this approach is illustrated in Fig. 5.11 where dash lines are previous results using LCNN and indicates performance improvement over LCNN.

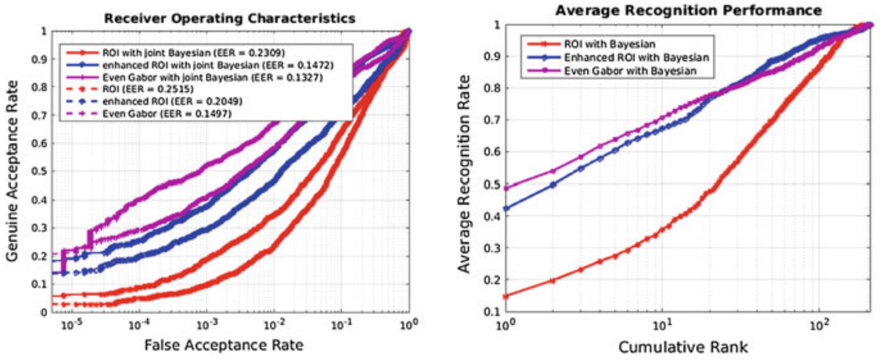


Fig. 5.11 Comparative ROC (*left*) and CMC (*right*) performance using LCNN with Joint Bayesian

5.5.5 Comparisons and Results Using Supervised Discrete Hashing

The supervised discrete hashing (SDH) scheme detailed in Sect. 5.4 was also investigated for the performance improvement. Only *first* session data was employed for the training part and the used for generating binarized bits that were used for matching using Hamming distance. The results using the ROC and CMC in Fig. 5.12 illustrates *consistent* performance improvement with the usage of SDH and the trends in the usage of enhanced ROI images are also consistent with our earlier observations.

The LCNN trained with triplet similarity loss function was also employed and used with the SDH to evaluate the performance. We attempted to ascertain the performance with different number of bits. Higher number of bits for SDH can be generally expected to offer superior results, but requires more training time. It should be noted that this method is actually a second-step training, and tries to map features from

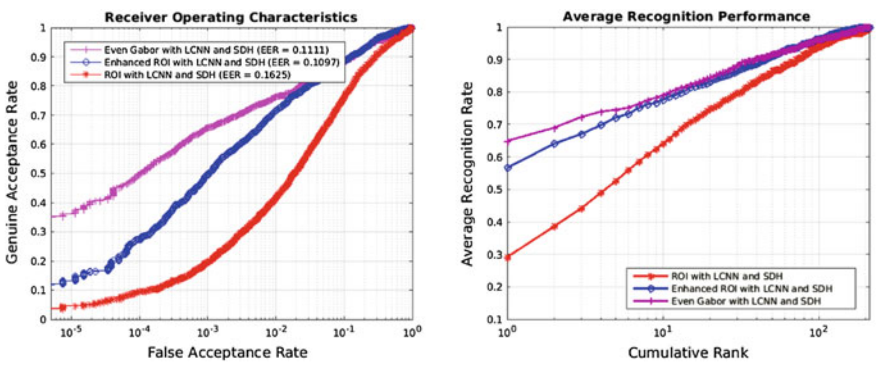


Fig. 5.12 Comparative ROC (*left*) and CMC (*right*) performance using LCNN with SDH

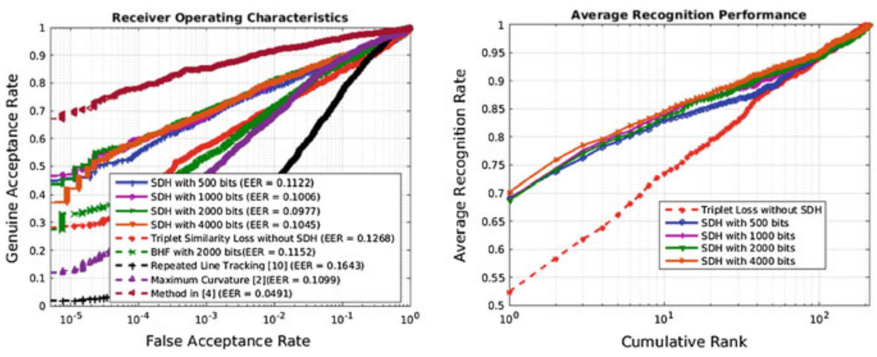


Fig. 5.13 Comparative ROC (*left*) and CMC (*right*) performance using SDH with triplet similarity loss function

the Euclidian space to the binary space. The training phase and hamming distance metrics can contribute to its superior performance. The combination of CNN with the hashing to reduce for the faster and real-time identification has also been attempted earlier [18] but for the face recognition problem. Authors in [18] introduced incorporated Boosted Hashing Forest (BHF) for the hashing and therefore we also attempted to incorporate BHF scheme to comparatively evaluate the performance. However, our results illustrated superiority of SDH over BHF and the template size using BHF was also fixed to 2000 bits. Although our results did not achieve significant improvement in the performance using BHF, its usage helps in remarkably reducing the template size. In order to ascertain comparative performance for matching finger vein images using the handcrafted features, we also performed additional experiments. The ROC from the same test images and matching protocols but using repeated line tracking [1] (also used as baseline in [21]) and curvatures [13] method is illustrated in Fig. 5.13. We can observe that the experimental results using SDH and LCNN offer superior performance and significantly reduced template size. Our results over the method using [8] are can be considered as competing and not yet superior but offers significantly reduced template size ($\sim$26 times smaller) over the best of the methods in [8] which is still the state-of-the-art method to date.

5.5.6 Results Using Modified VGG-16

The experimental results using modified VGG-16, as detailed in Sect. 5.3.3 are presented in Fig. 5.14. It should be noted that this CNN architecture generates single match score and therefore we cannot use SDH scheme to the infer features. We can infer from the ROCs in Fig. 5.14, that modified VGG-16 architecture generates superior performance for matching finger vein images as compared to the network trained using LCNN. This architecture directly generates the matching scores and

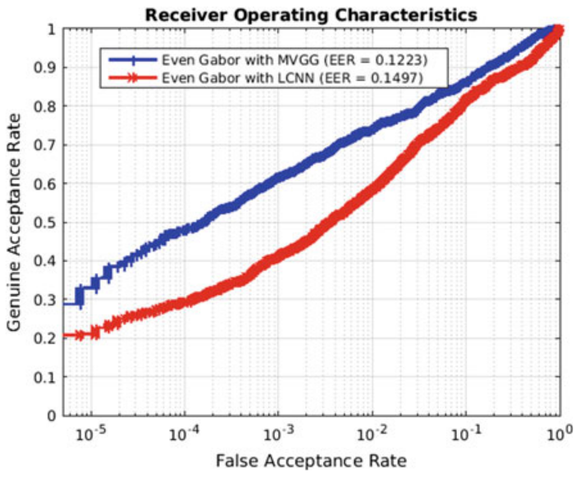


Fig. 5.14 Comparative performance using SDH with MVGG

therefore can minimize the problems from the inappropriate choice of distance metric (or weighting of features), which may be a convincible reason for its superior performance/results. It should however be noted that different training of the network can lead to variations in the results.

5.5.7 Results Using Single Fingers and Unregistered Fingers

Since we used both finger vein images from two fingers to form larger dataset (as in [8]) for above experiments, it is judicious to ascertain the performance when only one, i.e., index or middle, finger vein images are used in the training and test phase. The results using respective finger vein images from 105 different subjects are comparatively shown in Fig. 5.15 using the ROC. The performance using the images from both fingers (using 210 class formed by combination of index and middle finger for 105 different subjects) is superior to single finger, and index finger shows better performance than middle finger. Similar trends are also observed in [8] and can be attributed to the nature of dataset.

In earlier experiments, the first session data had images acquired from the same subjects who were providing their images during the second session and were used as test set for the performance. In order to ascertain robustness of self-learned features using the best scheme so far, we also evaluated the performance from the 51 subjects in this dataset which did not have any two-session finger vein images Therefore, images from none of these subject's images were employed during the training for CNN in any of the earlier experiments. The six images from these 51 subjects were used to ascertain performance using challenging protocol, i.e., all-to-all, so that we generated a total of 1530 genuine scores and 185436 impostor scores to ascertain such performance. The ROC corresponding to this independent test subjects finger

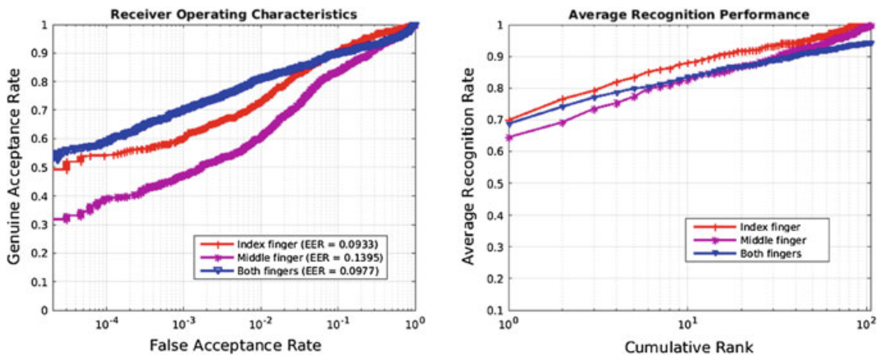
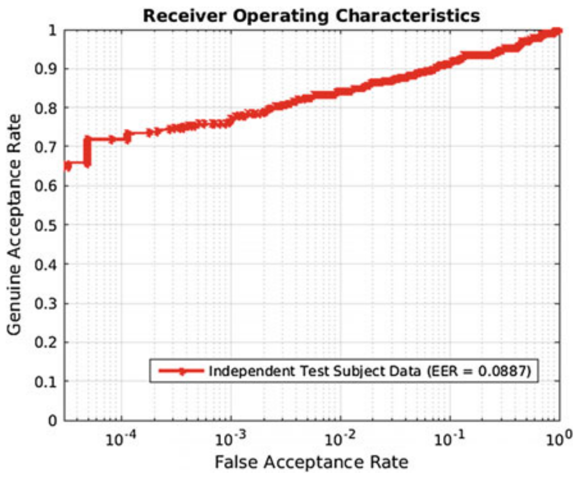


Fig. 5.15 Comparative ROC (*left*) and CMC (*right*) performance using SDH with triplet similarity loss experimented on single finger

Fig. 5.16 The performance using independent test subjects in [17] for matching finger vein images



vein data is shown in Fig. 5.16 and the results indicate promising performance from the self-learned features using a model trained (in Sect. 5.3.2 and SDH) for matching finger vein images from unknown subjects.

5.6 Discussion

This chapter has investigated finger vein matching performance using various convolutional neural network architectures. Unlike earlier work on finger vein image matching which largely employed handcrafted features, our emphasis has been to investigate automatically learned features using the capabilities of deep learning. We systematically investigated the performance improvement using just the ROI images and the systematically enhanced images that mainly emphasizes on subsurface vascular network. Our results consistently indicate superior performance from the CNN that are trained with images which have such enhanced vascular features.

According to our experimental results in Sect. 5.5.6, modified VGG-16 (MVGG) achieves superior performance than LCNN. However, MVGG requires significantly higher time for the training (also for the test phase). This can be largely attributed to the fact that it directly generates the match scores and therefore the loss function outputs propagate iteratively through the whole network to ascertain the similarity between a pair of finger vein images. At the same time, we cannot incorporate SDH (hashing scheme) with the MVGG, due to nonavailability of intermediate features, while the usage of SDH has shown to offer remarkable improvement in the performance.

It should be noted that the triplet similarity loss function helps to significantly improve the experimental performance using the LCNN. However, this approach cannot adequately make use of the label information, because it attempts to decrease the feature similarity between the pairwise images from the same subject, but cannot accurately locate the labels, i.e., identity of the subjects they are associated with. Supervised discrete hashing approach significantly improves performance and the retrieval speed, and decrease the storage which requires only 250 bytes (2000 bits) for the one template (feature vector). However, it should also be noted that this method needs a separate training phase and training time rapidly increases when the bit length or number of features are increased.

The work detailed in this chapter also had several constraints and therefore should be considered only preliminary. The database employed, although one of the largest two-session finger vein databases available in public domain, is still of smaller size for the deep learning based algorithms. There are several references in the literature that have shown promising performance but yet to demonstrate superior matching performance over the method in [8] using fair comparison or the same matching protocols. Therefore, we are justified in using the performance from [8], for this publicly available dataset, as the reference.

5.7 Conclusions and Further Work

This chapter has systematically investigated finger vein identification using the various CNN architectures. Unlike earlier work on finger vein matching which employed handcrafted features, our emphasis has been to investigate performance from the automatically learned features using the capabilities of deep learning. We systematically investigated the performance improvement using the ROI finger vein images, and the enhances images and consistently observed that the usage of ROI images with enhanced vascular features and attenuation of background (noise) can significantly improve the performance. The factors that most influence the accuracy of matching finger vein images is the depth of the network, the pretraining and the data augmentation in terms of random crops and rotations.

The LCNN architecture detailed in Sect. 5.3.2 that uses triplet similarity loss function achieves superior performance than those from the original *softmax* loss. Our experimental results using SDH illustrates that this hashing method significantly improves the matching performance and offers better alternative than BHF for the finger vein identification problem. Apart from two-session identification on same subjects, we also experimented on the datasets from subjects whose data was never available for training the CNN and this remarkable performance indicates high generalization capability for the finger vein identification problem. Our proposal for finger vein identification detailed in this chapter achieves smallest template size than using any other methods available in the literature to date. This work however had several constraints and therefore should be considered only preliminary. The database employed, although the largest two session finger vein images database available in public domain, is still of smaller size for the deep learning algorithms. Despite promising improvements in the accuracy from the publicly available (limited) training data, more work needs to be done to achieve significantly superior results than using the best performing method in [8]. Further work should use larger training dataset but should provide performance using the independent test data or using the publicly available dataset [17] to achieve more accurate alternative for the automated finger vein identification.

References

1. D. Chen, X. Cao, L. Wang, F. Wen, J. Sun, Bayesian face revisited: a joint formulation, in *Proceedings of ECCV 2012* (2012)
2. L. Chen, J. Wang, S. Yang, H. He, A finger vein image-based personal identification system with self-adaptive illuminance control. IEEE Trans. Instrum. Meas. **66**(2), 294–304 (2017)
3. L. Dong, G. Yang, Y. Yin, F. Liu, X. Xi, Finger vein verification based on a personalized best patches map, in *Proceedings of 2nd IJCB 2014*, Tampa, Sep.–Oct. 2014
4. F. Hillerstorm, A. Kumar, R. Veldhuis, Generating and analyzing synthetic finger-vein images, in *Proceedings of BIOSIG 2014*, Darmstadt, Germany, September 2014
5. https://polyuit-my.sharepoint.com/personal/csajaykr_polyu_edu_hk/_layouts/15/guestaccess.aspx?docid=11d706337994749759a2cc64cb70b604a&authkey=AcDq15geZ7942-7owNQfmYQ
6. M. Kono, H. Ueki, S. Umemura, A new method for the identification of individuals by using of vein pattern matching of a finger, in *Proceedings of 5th Symposium on Pattern Measurement*, pp. 9–12 (*in Japanese*), Yamaguchi, Japan, 2000
7. M. Kono, H. Ueki, S. Umemura, Near-infrared finger vein patterns for personal identification. Appl. Opt. **41**(35), 7429–7436 (2002)
8. A. Kumar, Y. Zhou, Human identification using finger images. IEEE Trans. Image Process. **21**, 2228–2244 (2012)
9. C. Liu, Y.H. Kim, An efficient finger-vein extraction algorithm based on random forest regression with efficient local binary patterns, in *2016 IEEE ICIP* (2016)
10. Y. Lu, S. Yoon, D.S. Park, Finger vein identification system using two cameras. Electron. Lett. **50**(22), 1591–1593 (2014)
11. Y. Lu, S. Yoon, S.J. Xie, J. Yang, Z. Wang, D.S. Park, Efficient descriptor of histogram of salient edge orientation map for finger vein recognition. Optic. Soc. Am. **53**(20), 4585–4593 (2014)

12. N. Miura, A. Nagasaka, T. Miyatake, Feature extraction of finger-vein patterns based on repeated line tracking and its application to personal identification, in *Machine Vision & Applications*, pp. 194–203, Jul, 2004
13. N. Miura, A. Nagasaka, T. Miyatake, Extraction of finger-vein patterns using maximum curvature points in image profiles, in *Proceedings of IAPR Conference on Machine Vision and Applications*, pp. 347–350, Tsukuba Science City, May 2005
14. F. Schroff, D. Kalenichenko, J. Philbin, Facenet: a unified embedding for face recognition and clustering (2015), arXiv:1503.03832
15. F. Shen, C. Shen, W. Liu, H.T. Shen, Supervised Discrete Hashing, in *Proceedings of CVPR, 2015*, pp. 37-45, Boston, June 2015
16. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014), arXiv:1409.1556
17. The Hong Kong Polytechnic University Finger Image Database (Version 1.0) (2012), http://www.comp.polyu.edu.hk/~csajaykr/fvdatavase.htm
18. Y. Vizilter, V. Gorbatsevich, A. Vorotnikov, N. Kostromov, Real-time face identification via CNN and boosted hashing forest, in *Proceedings of CVPR 2016 Biometrics Workshop, CVPR'W 2016*, Las Vegas, June 2016
19. C. Xie and A. Kumar, 'Finger vein identification using convolutional neural networks', The Hong Kong Polytechnic University, Tech. Report No. COMP K-25, Jan 2017
20. X. Wu, R.He, Z. Sun, T. Tan, A light CNN for deep face representation with noisy labels (2016), arXiv:1151.02683v2
21. Y. Ye, L. Ni, H. Zheng, S. Liu, Y. Zhu, D. Zhang, W. Xiang, FVRC2016: the 2nd finger vein recognition competition, in *2016 ICB* (2016)
22. Y. Zhou, A. Kumar, Contactless palmvein identification using multiple representations, in *Proceedings of BTAS 2010*, Washington DC, USA, September 2010
23. Z. Zhao, A. Kumar, Accurate periocular recognition under less constrained environment using semantics-assisted convolutional neural network. IEEE Trans. Info. Forensics Secur. **12**(5), 1017–1030 (2017)

Chapter 6
Iris Segmentation Using Fully Convolutional Encoder–Decoder Networks

Ehsaneddin Jalilian and Andreas Uhl

Abstract As a considerable breakthrough in artificial intelligence, deep learning has gained great success in resolving key computer vision challenges. Accurate segmentation of the iris region in the eye image plays a vital role in efficient performance of iris recognition systems, as one of the most reliable systems used for biometric identification. In this chapter, as the first contribution, we consider the application of Fully Convolutional Encoder–Decoder Networks (FCEDNs) for iris segmentation. To this extent, we utilize three types of FCEDN architectures for segmentation of the iris in the images, obtained from five different datasets, acquired under different scenarios. Subsequently, we conduct performance analysis, evaluation, and comparison of these three networks for iris segmentation. Furthermore, and as the second contribution, in order to subsidize the true evaluation of the proposed networks, we apply a selection of conventional (non-CNN) iris segmentation algorithms on the same datasets, and similarly evaluate their performances. The results then get compared against those obtained from the FCEDNs. Based on the results, the proposed networks achieve superior performance over all other algorithms, on all datasets.

6.1 Introduction

Deep learning techniques and convolutional neural networks (CNNs), in specific, are driving advances in artificial intelligence, as powerful visual recognition, classification and segmentation tools. Iris recognition is one of the most reliable and accurate biometric technologies used for human identification and authentication. The iris encloses many unique features, which make it a good candidate for distinguishing one person from another. Primary, the trabecular meshwork of the iris tissue is not

E. Jalilian (✉) · A. Uhl
Department of Computer Sciences, University of Salzburg, Jakob Haringer Str. 2, 5020 Salzburg, Austria
e-mail: ejalilian@cosy.sbg.ac.at

A. Uhl
e-mail: uhl@cosy.sbg.ac.at

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*, Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_6

genetically influenced during its development, has in excess of “266 degrees of freedom,” [6] and it is protected behind the eyelid, cornea and aqueous humour. Like any other biometrics system, the performance of iris recognition systems is highly depended on the accurate segmentation of the target region (iris) from the rest of image [31, 51]. And, to a great extent, the success or failure of the entire iris recognition system is considered to be directly dependent on the precision of this stage [5, 17].

6.1.1 Conventional (Non-CNN) Iris Segmentation Methods

Over the past decades, a lot of conventional (non-CCN) methods are proposed for iris segmentation. A quick review of related literatures unveils a significant number of these methods, which in turn enjoy versatile capabilities in iris segmentation [1, 7, 26, 36, 44, 47, 50]. In general, segmentation methods can be roughly classified into two main categories: contour-based methods and texture-based methods. The most well known contour-based methods are based on integro-differential operators [8], and Hough transforms [49]. The principles of integro-differential algorithms are based on search to find the largest difference of intensity over a parameter space, which normally corresponds to pupil and iris boundaries. Hough transform methods, however, try to find optimal circle parameters by exploring binary edge maps.

Performance of these methods is highly dependent on the images’ clear contour and the boundary contrast. Often, in normal conditions, limbic or pupillary boundaries in the images are of low-contrast, or may be of non-circular shape. In addition, the occlusions and specular reflections may introduce further contrast defects to the images. While plenty of improvement such as occlusion detection [18, 21], circle model improvement [29, 45], deformation correction [10], noise reduction [23], boundary fitting [47], and many further methods are introduced to compensate for such defects, yet due to their global approach in segmentation, the performance of these methods can be undermined by these defects, or even in some cases, they may result in total failure of system.

On the other hand, texture-based methods exploit the individual pixel’s visual aspects information, such as intensity, color, etc. to classify the iris pixels from the rest of image. The most promising methods in this category use some commonly known pixel-wise image classifiers such as: support vector machines (SVMs) [39], Neural networks [4], and Gabor filters [34] to classify iris pixels from the rest of image. In spite of the efforts to improve the performance of these group of algorithms [26, 43], yet these methods similarly suffer from the same group of defects such as diffusion, reflection and occlusion.

6.1.2 Convolutional Neural Networks (CNNs) for Segmentation

Convolutional neural networks (CNNs) have dramatically improved the state of the art in many image interpretation and analysis tasks, such as image segmentation and classification, since their early introduction [24, 25]. The principle of convolutional neural networks is based on hierarchical explanatory factors, where higher level, more abstract, concepts are learned from the lower level ones, with the help of convolutional operators. Assuming A_{ij} as the data vector at location (i, j) in a particular layer, and B_{ij} for the following layer, the output B_{ij} is computed as:

$$B_{ij} = f_{\alpha\beta}(\{A_{\beta_i+\delta_i,\beta_j+\delta_j}\} \; 0 \leq_{i,j} \leq \alpha)$$

where α is the convolutional kernel size, β is the stride or sub-sampling factor, and $f_{\alpha\beta}$ specifies the layer type. As already mentioned, the core building blocks of CNNs are the convolutional layers (Conv). The convolutional layers' parameters consist of a set of learnable filters. Each filter convolves the input volume and computes the dot product between the entries of the filter and the input at any position, and produces an activation map that gives the responses of that filter at every spatial position. The output of this layer then can be further processed by additional layers such as: nonlinear down-sampling layer (Pool), non-saturating activation function layer (ReLU), and further layers depending on the networks' architectures. All together these layers form the networks' encoder compartment. In addition to the encoding compartment, each network includes a decoding mechanism, in which the main task of labelling is performed, and the network's output is delivered as the classification scores.

The initial versions of convolutional neural networks were developed for classification tasks, where the networks' output was a single class label [22, 41]. In this type of networks, usually a fully connected layer (FC), as an inner-product function, was used to generate the classification scores. However, in many image processing tasks, as in iris segmentation, pixel-wise labelling of the region of interest was required. Prior approaches for pixel-wise segmentation used convents [14, 15]. In more recent CNNs' architectures, including the Fully Convolutional Neural Networks (FCNNs), usually a learnable upsampling layer (Upsample) is used to retrieve the feature maps, and then a softmax layer (Softmax), which normally computes the multinomial logistic loss of the softmax of its inputs, is employed to generate the classification scores. There exist various methods for enabling such upsampling mechanism [30, 46].

In Convolutional Encoder–Decoder Networks (CEDNs) the encoding mechanism, already explained, is repeated in the reverse mode to upsample the low-resolution output maps of the encoder network to full input resolution features of the input volume. Likewise, various methods are proposed for the upsampling mechanism in this type of networks. While some used switch codes to remap the features [32], others, like Ronneberger et al. [40], simply used cropped feature concatenation to generate the upsampled maps. In addition to using a primitive approach for upsampling, the network proposed by Ronneberger et al. is not fully convolutional, as the input and the output volumes do not match. The proposed mechanism to retrieve the full input volume (Overlap-tile) affects the training process anyway, and introduces further pre and postprocessing work load. The idea of using max-pooling indices from the corresponding encoder feature maps to generate the upsampled feature maps in the decoder network, "without learning," and then convolving with trainable filter banks, is recently proposed in a work by Badrinarayanan et al. [2]. The key leverage of applying such technique is improving the boundary delineation and preserving the edges with more precision.

6.2 CNNs in Iris Recognition

Nevertheless, when it comes to the application of deep learning techniques for iris segmentation, to the best of our knowledge, there exists only one proposal work on application of convolutional neural networks for iris segmentation. In [27] authors proposed two CNN-based iris segmentation models, namely: Hierarchical convolutional neural network (HCNN), and multi-scale fully convolutional network (MFCN). The former network is a simple CNN network composed of three blocks of alternative Conv and Pool layers, whose outputs are fed directly into a FC layer. The latter network includes six blocks of interconnected alternative Conv and Pool layers, whose outputs are simply fused through a single multiplication layer and then fed into a Softmax layer. The proposed networks are used for segmenting noisy-distanced iris images acquired from the Ubiris.v2,[1] and the Casia-distance-v4 databases.[2] Authors used a subset (500 and 400 images respectively) of these two databases. For the evaluation, authors referenced average error rates from other works, without carrying out direct experimental analysis on the same databases. Therefore, their results have to be considered with care, when it comes to the fair evaluation of segmentation accuracy. The ground-truth masks for the second database are manually labelled by the authors.

In [13], authors introduced two convolutional neural networks for iris recognition. But, these are proposed for "iris representation," not for segmentation. The networks are named "DeepIrisNet-A," which is based on the standard convolutional layers, and "DeepIrisNet-B," which uses a stack of so called "inception layers". Based on the results, the networks can model the micro-structures of iris well, and primarily outperform strong baseline based on descriptor and generalize well to new datasets.

In [35], authors used deep sparse filtering technique for iris feature extraction in a smart-phone based visible iris recognition system. In this work two layers of sparse filters are trained with 256 filters, with 16×16 kernels, to generate the feature

[1]Soft Computing and Image Analysis Lab, Univ. of Beira Interior, UBIRIS.v2 Dataset, see http://iris.di.ubi.pt/ubiris1.html.

[2]The Center of Biometrics and Security Research, CASIA Iris Image Database, see http://biometrics.idealtest.org.

maps. The final feature vector is formed by concatenating the histograms of these feature maps. They tested their system on many smart-phones and, depending on the smart-phone type, achieved different levels of accuracies. In [28] authors proposed a CNN, called "DeepIris," for heterogeneous iris verification, which learns relational features to measure the similarity between pairs of iris images. The network's architecture includes a pairwise filter layer, and a stock of alternative Pool, Conv, and Normalization (NR) layers, to generate a similarity map between two input images. The similarity measure is calculated as a scores with help of a FC layer at the end. Their results show that the proposed method achieves promising performance for both cross-resolution and cross-sensor iris verification.

6.3 A Fully Convolutional Encoder–Decoder Network

The architecture of the FCEDN networks used by us is similar to the work by Badrinarayanan et al. [2]. However, here we redesigned the Softmax layer to classify the outputs only into two classes (iris, and non-iris). Basically, this architecture proposes a fully convolutional encoder–decoder network, representing a core segmentation engine for pixel-wise semantic segmentation [2].

The core segmentation engine includes a 44-layered encoder network, and the corresponding decoder network. The encoder network's architecture is organized in five stocks. Each stock is comprised of a set of blocks, whose architectures are formed by a Conv layer, which convolves the input image with a kernel to produce the inputs' feature maps, followed by a batch normalized layer (BN), to normalize the layer input and avoid the "internal covariate shift" [19], and an element-wise rectified-linear nonlinearity layer (ReLU), as an activation function. The blocks then end up in a Pool layer (with a 2×2 window and stride 2), which applies nonlinear downsampling to the input and achieves translation invariance over small spatial shifts. While applying several Pool layers can help to obtain robust translation invariance, yet applying each layer leads to the loss of spatial resolution, specially in the boundary regions. This issue is resolved by storing max pooling indices, which are latter used for upsampling in the decoding network, in these networks.

The corresponding decoder network, likewise, has a 44-layered structure, which encompasses five stocks. Similarly, each stock is comprised of a set of blocks, whose architectures are formed from an Upsample layer, and trainable banks of decoder filters, including Conv, BN and ReLU layers, to generate the dense feature maps. As already specified, the Upsample layer uses max-pooling indices from the corresponding encoder feature maps to generate the upsampled feature maps, without learning and then the filter banks convolve the maps. Finally, the results are fed into a trainable Softmax layer (SoftmaxWithLoss). The Softmax layer classifies each pixel independently, so that the output of this layer is a N channel image of probabilities, where N is the number of classes. The output segmentation corresponds to the class with highest probability at each pixel. Applying this technique directly results in improvement of the boundary delineations and preservation of the edges. At the same time, this technique reduces the number of parameters, enabling end-to-end training.

6.3.1 Basic Variant

As we specified in the previous section, the overall structure of the network is organized as set of blocks, which in fact represent the network's learning units. Such a distinct unified structure expedites the modification of the Original network's architecture to best fit the required segmentation tasks. To this extent, and with the primary goal of facilitating the network analysis, an abstract variation of the Original network called "Basic" was introduced [2]. Using such an abstract architecture enables us to evaluate the effect of using less learning, and as the result, less convolutional and down-sampling units in the segmentation process. At the same time, such an architecture can offer faster and less computational expensive segmentation capabilities also. The overall network architecture, similarly, is composed of an encoder and the corresponding decoder networks. The encoder comprises four stocks, whose structures are similar to the Original network's blocks, incorporating Conv, BN, ReLU, and Pool layers. The decoder network's blocks, as well, include Upsample, Conv, and BN layers. The decoder network finally ends up to a SoftmaxWithloss layer.

6.3.2 Bayesian Variant

As another extension to the Original network, and as an attempt to propose a probabilistic pixel-wise segmentation model based on the deep learning technique, another variation of the Original network called "Baysian" was developed [20]. This network enables the probabilistic pixel-wise segmentation using Monte-Carlo sampling and the drop-out technique [12, 42]. The aim here is to find the posterior distribution over the convolutional weights w, given the observed training data x and the labels z.

$$p(w|x, z)$$

In practice, such a posterior distribution can only be approximated, for example, with variational inference techniques, such as minimization of the Kullback–Leibler (kl) divergence between the desired approximated distribution and the full posterior distribution [9].

$$kl(q(w) \mid\mid p(w|x, z))$$

Gal et al. [11] have already shown that minimizing the cross entropy loss objective function pretends to minimizing the Kullback–Leibler divergence term. Accordingly, they have proved that training neural networks with the stochastic gradient descent will promote the model to learn the distribution of weights, while avoiding

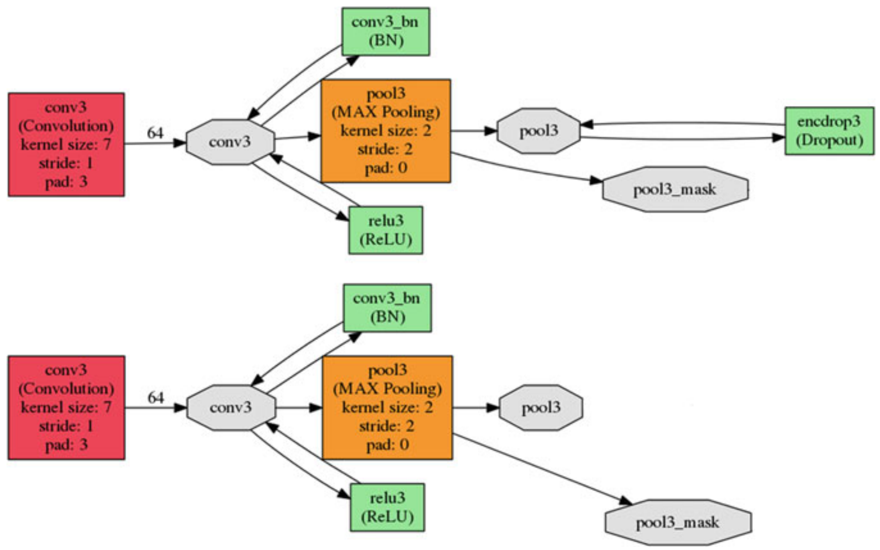

conv3 (Convolution) kernel size: 7 stride: 1 pad: 3
64
conv3
conv3_bn (BN)
pool3 (MAX Pooling) kernel size: 2 stride: 2 pad: 0
relu3 (ReLU)
pool3
encdrop3 (Dropout)
pool3_mask
conv3 (Convolution) kernel size: 7 stride: 1 pad: 3
64
conv3
conv3_bn (BN)
pool3 (MAX Pooling) kernel size: 2 stride: 2 pad: 0
relu3 (ReLU)
pool3
pool3_mask

Fig. 6.1 Architectural difference of the third block in the Bayesian-Basic (*up*), and the Basic (*down*) networks (Images are generated using Caffe framework)

over-fitting. Using the same technique, the overall architecture for the "Bayesian-Basic" network would be the same as the Basic network, except for this network includes extra drop-out layers, which are added to the two last blocks of the encoder, and the first two blocks of the decoder networks, as demonstrated in Fig. 6.1. Using this architecture, the posterior distribution over the weights would be sampled at the test time to generate the distribution of softmax class probabilities. The mean of this samples are taken for the segmentation prediction, and the variance is used to output the model uncertainty for each class. Monte-Carlo sampling is used for this purpose as it enables the qualitative understanding of the model uncertainty.

6.4 Experimental Framework

The main objective of this research is the application of FCEDNs for iris segmentation, and subsequently, providing evaluation and analysis of these networks' performance for different scenarios. For these purpose, after design and implementation of networks, each network was run on five different iris datasets, containing images acquired under different scenarios. The detailed specifications of these datasets are discussed in Sect. 4.1. The segmentation capability of each network then was evaluated and analysed with the help of three different segmentation scores, whose details are specified in Sect. 4.2. Next, in order to facilitate true assessment of the performance of networks for iris segmentation, a collection of conventional iris segmentation algorithms, whose details are specified in Sect. 6.6, got run on the same datasets' testing subsets, and their corresponding performance analysis and evaluation was presented, and compared against those of the FCEDNs.

6.4.1 Datasets

In this work we have selected five well-known iris datasets. The datasets were selected to include versatile iris image acquisition scenarios. The Notredame dataset (subset of ND-Iris-0405 database[3]) includes 835 iris images of 30 different subjects. The image acquisition was done in near-infrared spectrum, in an indoor environment, with the LG 2200 iris biometric system. For our experiments we used 670 images (24 subjects) for the training, and 165 images (six subjects) for the testing of the networks. The Casia-iris-interval-v4 dataset[4] contains a total of 2640 iris images belonging to 249 subjects. Images are acquired under near-infrared illumination, with a close-up iris camera. For our experiments 1307 instances of the right eye images were used, out of which 1045 images (192 subjects) were used for the training, and 262 image (57 subjects) were used for the testing of the networks.

The IITD database[5] consists 1120 iris images corresponding to 224 subjects. All these images are acquired in the indoor environment, with the Jiris, Jpc1000 digital CMOS camera in near-infrared spectrum. For our experiments 900 images (180 subjects) were used for the training, and 220 images (44 subjects) were used for the testing of the networks. The Ubiris dataset (subset of the Ubiris.v2 database) contains 2250 iris images, from 100 different subjects. The images were acquired with a Nikon E5700 camera and split into two parts. The first part includes iris images taken under controlled condition, simulating the enrolment stage. The second part includes iris images which are captured under real-world setup, with natural luminosity corresponding heterogeneity in reflections, contrast and focus. The dataset also contains off-angle iris images captured from various distances with occlusions. For our experiments we used 2055 images of this dataset for the training and 225 images for the testing of the networks.

And finally, the Casia-iris-ageing-v5 dataset[6] contains 120 images per eye and user from video sequences captured in 2009, and 20 images per eye and user from video sequences captured in 2013. For our experiments we used total of 1880 images of both eyes of 94 users from both sessions. Out of that, 1500 images of 75 users were used for the training, and 380 images, corresponding to 19 users, were used for the

[3]Computer Vision Research Lab, Univ. of Notre Dame, Iris Dataset 0405, see https://sites.google.com/a/nd.edu/public-cvrl/data-sets.

[4]The Center of Biometrics and Security Research, CASIA Iris Image Database, see http://biometrics.idealtest.org.

[5]Indian Institute of Technology Delhi, IIT Delhi Iris Database, see http://www4.comp.polyu.edu.hk/~csajaykr/database.php.

[6]See http://www.biometrics.idealtest.org.

testing of the networks. Special attention should be paid to the fact that the selection of the datasets was subject to availability of the ground-truth masks required for the training process. For this work the ground-truth masks were acquired from the Irisseg-ep database provided by WaveLab of the University of Salzburg [16]. The selection of the training and testing subsets followed the Pareto principle, where of total instances, approximately 80% of the data was used for training and 20% for testing, while subjects are not overlapped, and no instances are included in other's set. Also in order to maintain fair spatial input to the networks, images in all datasets was resized to 480×360.

6.4.2 Metrics and Measurements

In order to facilitate holistic statistical analysis and proper assessment of the capabilities of the FCEDNs and the other conventional algorithms on iris segmentation, we have considered a set of evaluation metrics, which cover key segmentation measures such as: true positives (tp), false negatives (fn), and false positive (fp). To this extent, primarily the NICE.I protocol, which is widely accepted for evaluation of iris segmentation accuracy, got adapted. The segmentation error score nice1 calculates the proportion of corresponding disagreeing pixels (by the logical exclusive-or operator) over all the image as follows:

$$nice1 = \frac{1}{c \times r} \sum_{c'} \sum_{r'} O(c', r') \otimes C(c', r') \tag{6.1}$$

where c and r are the columns and rows of the segmentation masks, and $O(c', r')$ and $C(c', r')$ are, respectively, pixels of the output and the ground-truth mask. The second segmentation error score intends to compensate the disproportion between the priori probabilities of iris and non-iris pixels in the images. The type-I and type-II error score nice2 averages between the (fp) and (fn) rates as follow

$$nice2 = \frac{1}{2}\,(fp + fn) \tag{6.2}$$

The values of nice1 and nice2 are bounded in the [0, 1] interval, and in this context, 1 and 0 are respectively the worst and the optimal values. Additionally, in order to provide comprehensive synopsis of the networks' performances, three more standard measures of segmentation accuracy were considered, namely: precision, recall, and f1 measure, which are well-known measures in the field of information retrieval [38]. Precision gives the percentage of retrieved iris pixels which are correctly segmented as follows:

$$p = \frac{tp}{tp + fp} \tag{6.3}$$

Alternatively, recall provides the same measure using false negatives as follows:

$$r = \frac{tp}{tp + fn} \tag{6.4}$$

Last but not least, the f1 measure is the harmonic mean of p and r, and is calculated as follows:

$$f1 = \frac{2rp}{r + p} \tag{6.5}$$

The values for these three measures are bounded in the [1, 0] interval, and in this context, 0 and 1 are the worst and the optimal values respectively.

6.4.3 Network Implementation

In the first step, we implemented the FCEDNs on the "Caffe" deep learning framework. Caffe is one of the most favourited deep learning frameworks, which at its core, is written in c++. Models and optimizations are defined by configuration without hard-coding, which in turn accelerate the training and testing process. Switching between CPU and GPU can be done just by setting a single flag, and the interface is extensible to python (Pycaffe) and matlab (Matcaffe). Network architecture is defined in separate "prototxt" files, and training and testing parameters are defined in another similar file called "solver". The prototxt files get loaded during the training and testing using caffe commands.

Architectural implementation and the technical specification for the Original, Basic, and Bayesian-Basic networks are presented in Tables 6.1, 6.2, and 6.3 respectively. The convolutional kernel size for the Original network was set to 3×3, and in order to provide a wide context for smooth labelling, this value was set to 7×7 for both of the Basic networks.

We trained our networks by Stochastic Gradient Descent (SGD) back propagation algorithm

$$U_{t+1} = \mu U_t - \alpha \nabla L(W) \tag{6.6}$$

$$W_{t+1} = W_t + U_{t+1} \tag{6.7}$$

Formally, at each iteration $t + 1$ the SGD algorithm computes the update value U_{t+1} and the updated weights W_{t+1}, given the previous weight update U_t and current weights W_t. The algorithm updates the weights W by linear combination of the negative gradient $\nabla L(W)$ and the previous weight update U_t. The learning rate α is the weight of the negative gradient, and the momentum μ is the weight of the previous update.

For our experiments, the momentum value for the SGD algorithm was set to 0.9. In order to investigate the networks' training process, we have considered two

Table 6.1 Architecture and specification of the original encoder (left)–decoder (right) network

Layer	Information	Layer	Layer	Layer	Information	Layer	Layer
Convolution	Output	Batch-Normalization	ReLU	Convolution	Output	Batch-Normalization	ReLU
Conv1-1	64	Conv1-1-b	Rlu1-1	Conv5-3-D	512	Conv5-3-D-b	Rlu5-3-D-b
Conv1-2	64	Conv1-2-b	Rlu1-2	Conv5-2-D	512	Conv5-2-D-b	Rlu5-2-D-b
Conv2-1	128	Conv2-1-b	Rlu2-1	Conv5-1-D	512	Conv5-1-D-b	Rlu5-1-D-b
Conv2-2	128	Conv2-2-b	Rlu2-2	Conv4-3-D	512	Conv4-3-D-b	Rlu4-3-D-b
Conv3-1	256	Conv3-1-b	Rlu3-1	Conv4-2-D	512	Conv4-2-D-b	Rlu4-2-D-b
Conv3-2	256	Conv3-2-b	Rlu3-2	Conv4-1-D	256	Conv4-1-D-b	Rlu4-1-D-b
Conv3-3	256	Conv3-3-b	Rlu3-3	Conv3-3-D	256	Conv3-3-D-b	Rlu3-3-D-b
Conv4-1	512	Conv4-1-b	Rlu4-1	Conv3-2-D	256	Conv3-2-D-b	Rlu3-2-D-b
Conv4-2	512	Conv4-2-b	Rlu4-2	Conv3-1-D	128	Conv3-1-D-b	Rlu3-1-D-b
Conv4-3	512	Conv4-3-b	Rlu4-3	Conv2-2-D	128	Conv2-2-D-b	Rlu2-2-D-b
Conv5-1	512	Conv5-1-b	Rlu5-1	Conv2-1-D	64	Conv2-1-D-b	Rlu2-1-D-b
Conv5-2	512	Conv5-2-b	Rlu5-2	Conv1-2-D	64	Conv1-2-D-b	Rlu1-2-D-b
Conv5-3	512	Conv5-3-b	Rlu5-3	Conv1-1-D	2	Conv1-1-D-b	
Pooling	Stride	Kernel size		Upsample	Width	Height	Scale
Pool1	2	2		Upsamp5	30	23	2
Pool2	2	2		Upsamp4	60	45	2
Pool3	2	2		Upsamp3	–	–	2
Pool4	2	2		Upsamp2	–	–	2
Pool5	2	2		Upsamp1	–	–	2

Table 6.2 Architecture and specification of the basic encoder (left)–decoder (right) network

Layer	Information	Layer	Layer	Layer	Information	Layer
Convolution	Output	Batch-Normalization	ReLU	Convolution	Output	Batch-Normalization
Conv1	64	Conv1-b	Relu1	Conv4-D	64	Conv4-D-b
Conv2	64	Conv2-b	Relu2	Conv3-D	64	Conv3-D-b
Conv3	64	Conv3-b	Relu3	Conv2-D	64	Conv2-D-b
Conv4	64	Conv4-b	Relu4	Conv1-D	64	Conv1-D-b
				ConvC-D	2	
Pooling	Stride	Kernel size		Upsample	Scale	
pool1	2	2		Upsamp4	2	
pool2	2	2		Upsamp3	2	
pool3	2	2		Upsamp2	2	
pool4	2	2		Upsamp1	2	

Table 6.3 Architecture and specification of the Bayesian-Basic encoder (left)–decoder (right) network

Layer	Information	Layer	Layer	Layer	Information	Layer
Convolution	Output	Batch-Normalization	ReLU	Convolution	Output	Batch-Normalization
Conv1	64	Conv1-b	Relu1	Conv4-D	64	Conv4-D-b
Conv2	64	Conv2-b	Relu2	Conv3-D	64	Conv3-D-b
Conv3	64	Conv3-b	Relu3	Conv2-D	64	Conv2-D-b
Conv4	64	Conv4-b	Relu4	Conv1-D	64	Conv1-D-b
				ConvC-D	2	
Pooling	Stride	Kernel size		Upsample		Scale
pool1	2	2		Upsamp4		2
pool2	2	2		Upsamp3		2
pool3	2	2		Upsamp2		2
pool4	2	2		Upsamp1		2
Drop-out		Ratio		Drop-out		Ratio
encdrop3		0.5		D-drop4		0.5
encdrop4		0.5		D-drop3		0.5

learning rates for the different variations of the networks. For the Original network the learning rate was set to 0.001, and for the Basic networks this value was set to 0.1. The learning rates are set optimally based on the work by Badrinarayanan et al. [2]. The direct effect of such a small learning rate is slow but more stable convergence of the network.

Figure 6.2 clearly illustrates this effect on these two network architectures during the training process. Table 6.4 summarizes the training parameters, which were set

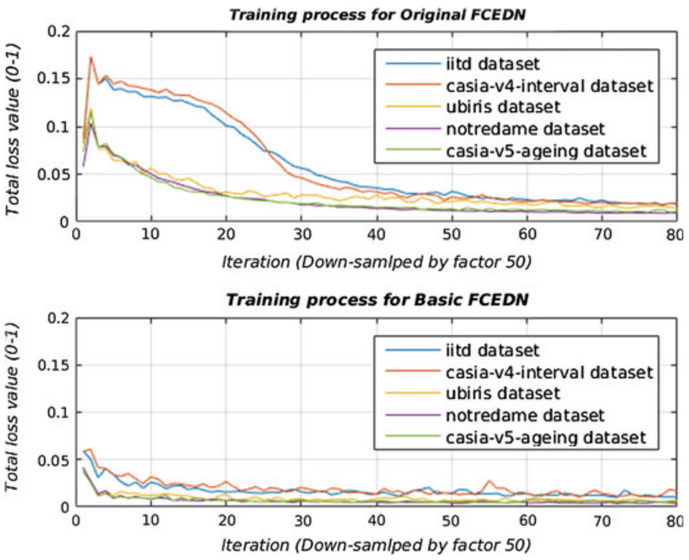


Fig. 6.2 Overall loss value in the first 4000 iterations of the training process for the Original (*up*), and the Basic (*down*) architectures on different datasets

Table 6.4 Solver parameters for the networks

Parameter	Original	Bayesian-Basic	Basic
Iterations	10,000	10,000	10,000
Momentum	0.9	0.9	0.9
Learning Rate	0.001	0.1	0.1

in the solver files. The setting criteria for these parameters is investigated in the work of Leon Bottou in this regard [3].

In our experiments the networks were trained per single image till the convergence point, where the overall loss for both segmentation classes decreased to less than 4%.

6.5 The First Segmentation Experiment Results

We have presented the segmentation results of the FCEDNs in Tables 6.5 and 6.6, as average segmentation scores per iris dataset, and per network respectively. As Table 6.5 demonstrates, the best results are obtained on the Notredame and the Casia5a datasets, and the worst ones on the Ubiris dataset. This is simply due to the difficulty level of these datasets. On the other hand, as it can be seen in Table 6.6, the Bayesian-Basic network outperforms the other two networks, with lower (mean) μnice1, μnice2 and higher μf1 scores (0.0316, 0.0571 and 0.8985 respectively),

Table 6.5 Average FCEDNs' segmentation scores per dataset

FCEDN	Dataset	nice1	nice2	f1
Original	iitd	0.0591	0.0659	0.8661
	notredame	0.0213	0.0424	0.8617
	casia4i	0.0561	0.0588	0.8826
	ubiris	0.0342	0.1249	0.7691
	casia5a	0.0160	0.0420	0.8951
Basic	iitd	0.0539	0.0594	0.8892
	notredame	0.0107	0.0269	0.9351
	casia4i	0.0448	0.0438	0.9072
	ubiris	0.0423	0.1517	0.7700
	casia5a	0.0086	0.0261	0.9510
Bayesian-Basic	iitd	0.0682	0.0701	0.8489
	notredame	0.0095	0.0282	0.9426
	casia4i	0.0391	0.0407	0.9192
	ubiris	0.0306	0.1116	0.8407
	casia5a	0.0105	0.0351	0.9413

Table 6.6 Average FCEDNs segmentation scores per network

FCEDN	μ nice1	μ nice2	μ f1
Bayesian-Basic	0.0316	0.0571	0.8985
Basic	0.0321	0.0616	0.8905
Original	0.0373	0.0668	0.8549

on overall datasets. This is directly due to the probabilistic technique used in this network, and the results clearly endorse the enhanced segmentation capability of the Basic network after applying this technique. The Basic network has comparatively moderate performance on the iris datasets with mean sores of 0.0321, 0.0616, and 0.8905 for μnice1, μnice2, and μf1 respectively. This is principally due to the simple structure of this network, which relays on the appearance information from shallow, fine layers to produce segmentations.

The Original network comes at the end, with mean sores of 0.0373, 0.0668 and 0.8549 for μnice1, μnice2 and μf1 respectively. This is meanly due to the comparatively deep structure of this network, which combines semantic information from deep, coarse layers with appearance information from shallow, fine layers to produce segmentations.

Figure 6.3 demonstrates the networks' best and worst performances samples for different datasets. More detailed performance analysis of the networks for iris segmentation per dataset is presented in Fig. 6.4, which provides further statistical information such as: min, max, median, quantiles and outliers in the form of Box-plots. The outliers are classified based on the following mechanism, where q_3 and q_1 are

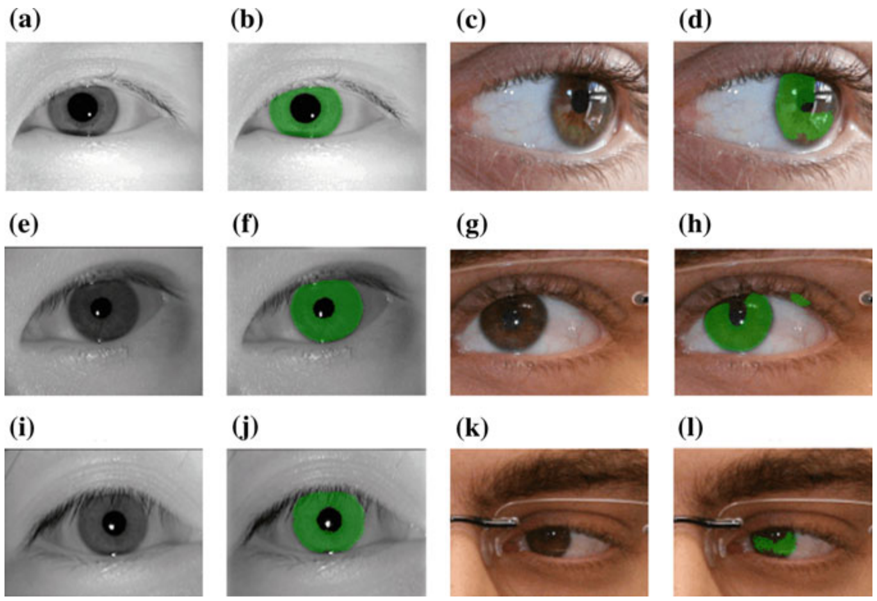


Fig. 6.3 The best (**b**) and the worst (**d**) performance samples of the Bayesian-Basic network on the notredame (**a**) and the ubiris (**c**) datasets' samples respectively. And the best (**f**) and the worst (**h**) performance samples of the Basic network on the casia5a (**e**) and the ubiris (**g**) datasets' samples respectively. And the best (**g**) and the worst (**l**) performance samples of the Original network on the casia5a (**i**) and the ubiris (**k**) datasets' samples respectively

the 25th and 75th percentiles of the scores, and w corresponds to approximately ± 2.7 of the standard deviation (σ): The scores are classified as outlier if they are greater than Z_1 or smaller than Z_2.

$$Z_1 = q_3 + w \times (q_3 - q_1) \tag{6.8}$$

$$Z_2 = q_3 - w \times (q_3 - q_1) \tag{6.9}$$

Furthermore, in order to assess how well the segmentation results generalize to the entire datasets, we trained and subsequently tested the networks on each dataset, applying the K-Fold cross-validation technique. For this purpose, we partitioned each dataset into five complementary subsets, and performed the training with four subsets, and validated the results on the remained subset. Likewise, five rounds of cross-validation were performed on each dataset separately without overlapping.

Figure 6.5 demonstrates the results for the cross-validation per segmentation score on the Notredame dataset. Table 6.7 demonstrates the average cross-validation results for all different datasets. As the results show, the average scores are quiet similar to those of the onefold experiments, and while most scores show around 2% difference, the maximum differences in the scores doe not exceed over 4%.

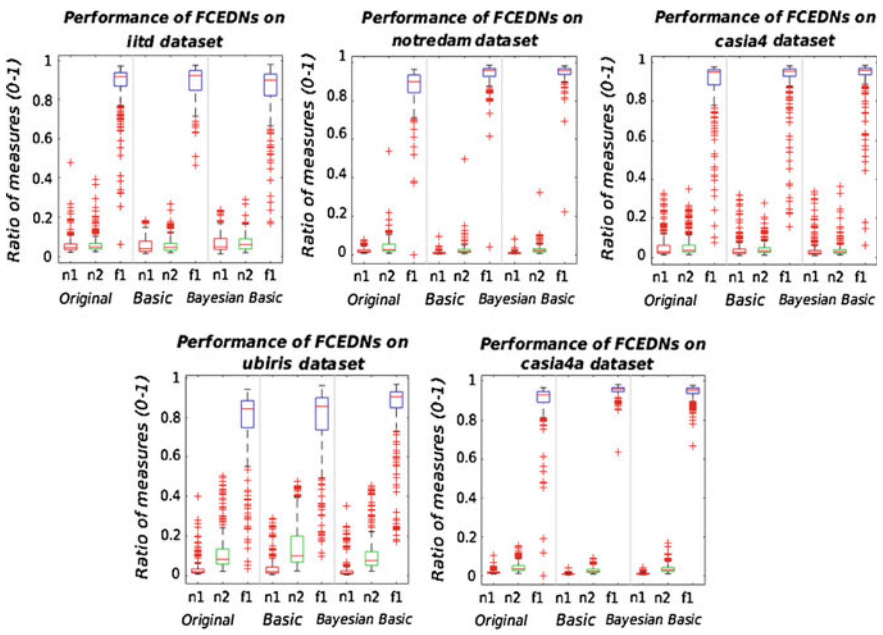


Fig. 6.4 Performance of the FCEDNs per dataset using segmentation errors: nice1 (n1), nice2 (n2) and f1 (f1)

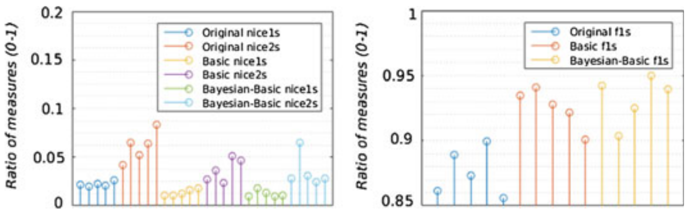


Fig. 6.5 Fivefold cross-validation results on the notredame dataset, demonstrating segmentation scores nic1, nice2 (*left*), and f1 (*right*) per round

Table 6.7 Average cross-validation results for all datasets

Network	Original			Basic			Bayesian-Basic		
Dataset	nice1	nice2	f1	nice1	nice2	f1	nice1	nice2	f1
Casia5a	0.0135	0.0392	0.9000	0.0197	0.0385	0.9250	0.0112	0.0332	0.9400
Casia4i	0.0415	0.0492	0.9175	0.0330	0.0382	0.9375	0.0412	0.0362	0.9250
Iitd	0.0365	0.0353	0.9400	0.0277	0.0322	0.9510	0.0292	0.0337	0.9500
Notredame	0.0220	0.0655	0.8775	0.0135	0.0387	0.9225	0.0132	0.0367	0.9300
Ubiris	0.0305	0.0898	0.7200	0.0262	0.0687	0.7900	0.0187	0.0675	0.8625

Table 6.8 Running time, per segmentation for FCEDNs

FCEDN	Original	Bayesian-Basic	Basic
Running time (s)	18.075	58.661	10.162

Furthermore, we measured the average running time, per segmentation, of different networks for a system with Intel-Xeon E5-1620 3.50 GHz cpu, and 32GiB memory. The results are presented in Table 6.8 respectively. Basically caffe is not optimized for the inlet processors. While based on the developers, using suitable GPUs and cuDNN, Caffe is considered as the fastest convent implementations available.[7]

6.6 The Second Segmentation Experiment Results

Next, in order to streamline the proper assessment of the capabilities of FCEDNs for iris segmentation, and to enable the comparative analysis of these networks' performance, a set of conventional iris segmentation methods (Convs) was considered to be run on the same datasets' testing subsets.

Osiris is an open-source iris recognition software, that includes an iris segmentation algorithm, which uses the Viterbi algorithm on the gradient map of anisotropic smoothed image for iris segmentation [33]. Caht (contrast-adjusted hough transform) [36], Wahet (weighted adaptive Hough and ellipsopolar transform) [47], and Ifpp (iterative Fourier-series push pull) [48] are further open-source iris segmentation algorithms used in this experiment, which are acquired from the Iris-Toolkit package provided by Wavelab at the University of Salzburg [37]. The performance of these algorithms was evaluated using the same segmentation scores used in the first experiment. The performance results of the conventional algorithms are represented in Tables 6.9 and 6.10, as average classification score per iris datasets and per algorithm respectively.

Generally, Osiris tends to underestimate the iris boundaries, as it tries to mask the obstructions out, leading to high precision but lower recall. However, Wahet and Caht lean to overestimate the iris boundaries, resulting in higher recall than precision. The reason for this is that these two algorithms do not utilize eyelid filters. Similarly, in the Ifpp algorithm, less pronounced boundaries are largely affected by noise or eyelids. Therefore, the less expressive boundaries are reconstructed from the more stable ones.

As it can be seem in Table 6.9 the algorithms perform better on less difficult datasets such as the Notredame and the Casia5a, while the worst results are obtained on the Ubiris dataset. In both cases the algorithms' performance results conform with the FCEDNs' results, which endorses the equity of the experimental scheme. As it

[7]Caffe Deep learning framework, see http://caffe.berkeleyvision.org/.

Table 6.9 Average convs' segmentation scores per dataset

Algorithm	Dataset	nice1	nice2	f1
Wahet	iitd	0.1377	0.1762	0.7337
	notredame	0.0248	0.0875	0.8619
	casia4i	0.0608	0.0842	0.8949
	ubiris	0.2743	0.4498	0.1977
	casia5a	0.0248	0.0836	0.8648
Caht	iitd	0.1138	0.1560	0.7767
	notredame	0.0361	0.1408	0.7941
	casia4i	0.1161	0.1470	0.7651
	ubiris	0.1226	0.4809	0.1048
	casia5a	0.0369	0.1514	0.7753
Ifpp	iitd	0.1142	0.1508	0.7965
	notredame	0.0294	0.1113	0.8359
	casia4i	0.1532	0.2372	0.6278
	ubiris	0.2379	0.3970	0.2899
	casia5a	0.0288	0.1123	0.8504
Osiris	iitd	0.0555	0.0757	0.8817
	notredame	0.0131	0.0231	0.9194
	casia4i	0.0565	0.0673	0.8862
	ubiris	0.1827	0.4095	0.2328
	casia5a	0.0181	0.0331	0.8917

Table 6.10 Average convs' segmentation scores per algorithm

Algorithm	μ nice1	μ nice2	μ f1
Osiris	0.0652	0.1217	0.7624
Caht	0.0851	0.2152	0.6432
Wahet	0.1045	0.1763	0.7106
Ifpp	0.1127	0.2017	0.6801

can be seen in Table 6.10 the Osiris algorithm outperforms the other threes, with lower (mean) μnice1, μnice2 and higher μf1 scores (0.0652, 0.1217 and 0.7624 respectively). Figure 6.6 provides further statistical information such as: min, max, median, quantiles, and outliers, about the segmentation performances of these algorithms per dataset in the form of Box-plots. Furthermore, we measured the average running time, per segmentation, of the conventional algorithms for the system specified previously. The results are presented in Table 6.11 respectively.

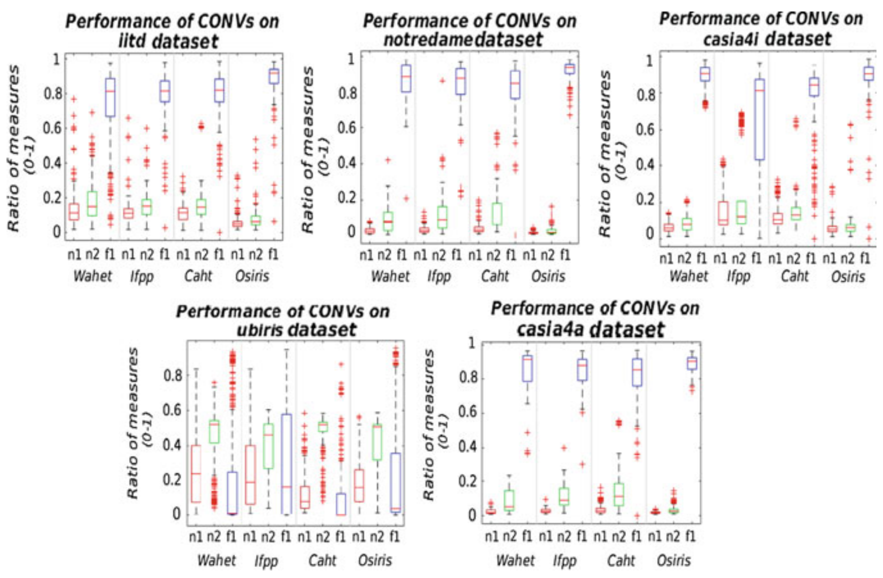


Fig. 6.6 Performance of the conventional algorithms per dataset using segmentation errors: nice1 (n1), nice2 (n2) and f1 (f1)

Table 6.11 Running time, per segmentation for convs

Algorithm	Osiris	Wahet	Caht	Ifpp
Running time (s)	0.583	0.602	6.730	0.470

6.7 Analysis and Discussion

Simple statistical comparison of the segmentation results of the FCEDNs with the conventional algorithms' results demonstrates the superiority of the FCEDNs for iris segmentation over the other conventional algorithms. As it can be seen in Tables 6.6 and 6.10, even the worst FCEDNs' performance result, which is shown by the Original network, scoring: 0.0373, 0.0668 and 0.8549 for nice1, nice2 and f1 respectively, is better then the best conventional algorithms' result, which is obtained by the Osiris algorithm scoring: 0.0652, 0.1217 and 0.7624 for nice1, nice2, and f1 respectively. Yet if we consider the best FCEDNs' result, which is obtained by the Bayesian-Basic network (0.0316, 0.0571, and 0.8985 for nice1, nice2 and f1 respectively), the prominence of the proposed FCEDNs over the conventional algorithms would be consolidated by power of two or three.

The greatest supremacy of the proposed FCEDNs is revealed when analysing the segmentation results per dataset in Tables 6.5 and 6.9. As it can be seen in the tables, the worst segmentation scores for almost all conventional algorithms and FCEDs are obtained on the Ubiris dataset, which deliberately contains samples of off-angle iris

Table 6.12 Average segmentation scores of all methods on the Ubiris dataset

Method	nice1	nice2	f1
Bayesian-Basic	0.0306	0.1116	0.8407
Original	0.0342	0.1249	0.7691
Basic	0.0423	0.1517	0.7700
Osiris	0.1827	0.4095	0.2328
Caht	0.1226	0.4809	0.1048
Wahet	0.2743	0.4498	0.1977
Ifpp	0.2379	0.3970	0.2899

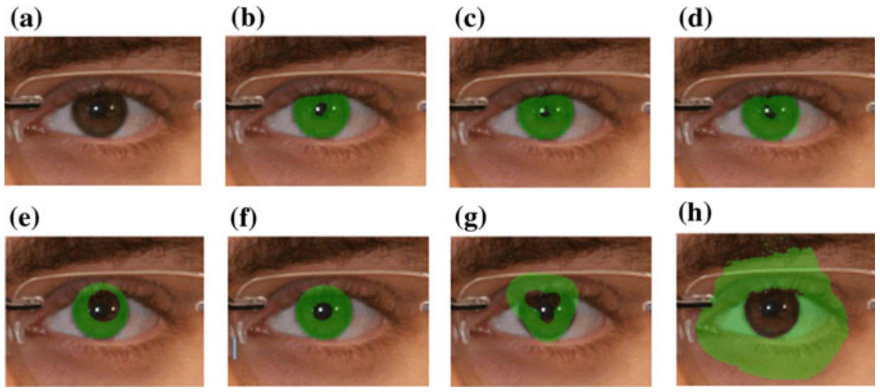


Fig. 6.7 A sample iris image with glasses (**a**) from the ubiris dataset versus the output segmentation masks of: Bayesian-Basic (**b**), Basic (**c**), Original (**d**), Caht (**e**), Wahet (**f**), Ifpp (**g**), and Osiris (**h**)

images recorded from various distances with different types of occlusions, including glasses. While most conventional algorithms such as Wahet, Caht and Ifpp even fail to satisfy the minimum segmentation scores, all FCEDNs demonstrate robust segmentation capabilities on such a difficult and divergent iris dataset.

This can be easily interpreted from Table 6.12, which summarizes the segmentation results of the conventional algorithms along with the FCEDNs on the Ubiris dataset. A simple visual comparison of the Box-plots for the Ubiris dataset in Figs. 6.4 and 6.6 demonstrates this fact clearly also. Figure 6.7 displays a sample iris image with glasses from the Ubiris dataset, along with the corresponding output masks of all segmentation methods (FCEDNs and Convs).

6.8 Conclusion

Accurate segmentation of the iris region from the rest of image plays a vital role in efficient performance of iris recognition systems, and success of the total system is considered to be directly related to the precision of this stage. In this work we have presented the application of deep learning techniques and FCEDNs for iris segmentation. To this extent, we applied three types of networks for iris segmentation. The performance of the networks was tested and evaluated on five different datasets. The evaluation was carried out using three popular segmentation error scores. Furthermore, in order to streamline proper assessment of the performance of the networks, we presented statistical analysis and the performance evaluation of four well-known conventional iris segmentation algorithms on the same datasets, and compared the results against those obtained from the networks. Results demonstrate the superiority of the networks for iris segmentation over all other algorithms. Yet the greatest supremacy of the proposed networks unveils when dealing with difficult iris images such as off-angle images recorded from various distances with different types of occlusions including glasses. In future work we plan to perform more application specific analysis on these types of networks, and at the same time carry out further research to optimize their design and architecture, and improve their performance for different segmentation tasks.

Acknowledgements This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 700259.

References

1. U. Andreas, W. Peter, Multi-stage visible wavelength and near infrared iris segmentation framework, in *Proceedings of the International Conference on Image Analysis and Recognition (ICIAR'12)*, LNCS (Aveiro, Portugal, 2012), pp. 1–10
2. V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: a deep convolutional encoder-decoder architecture for image segmentation (2015), arXiv:1511.00561
3. L. Bottou, Stochastic gradient descent tricks, in *Neural Networks: Tricks of the Trade* (Springer, 2012), pp. 421–436
4. R.P. Broussard, L.R. Kennell, D.L. Soldan, R.W. Ives, Using artificial neural networks and feature saliency techniques for improved iris segmentation, in *International Joint Conference on Neural Networks, 2007. IJCNN 2007* (IEEE, 2007), pp. 1283–1288
5. Y. Chen, M. Adjouadi, A. Barreto, N. Rishe, J. Andrian, A computational efficient iris extraction approach in unconstrained environments, in *IEEE 3rd International Conference on Biometrics: Theory, Applications, and Systems, 2009. BTAS'09* (IEEE, 2009), pp. 1–7
6. J. Daugman, Recognizing people by their iris patterns. Inf. Secur. Tech. Rep. **3**(1), 33–39 (1998)
7. J. Daugman, How iris recognition works. Int. Conf. Image Process. **1**, I-33–I-36 (2002)
8. J.G. Daugman, High confidence visual recognition of persons by a test of statistical independence. IEEE Trans. Pattern Anal. Mach. Intell. **15**(11), 1148–1161 (1993)
9. J. Denker, Y. Lecun, Transforming neural-net output levels to probability distributions, in *Proceedings of the 3rd International Conference on Neural Information Processing Systems* (Morgan Kaufmann Publishers Inc, 1990), pp. 853–859
10. V. Dorairaj, N.A. Schmid, G. Fahmy, Performance evaluation of non-ideal iris based recognition system implementing global ica encoding, in *IEEE International Conference on Image Processing 2005*, vol. 3 (IEEE, 2005), pp. III–285
11. Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with bernoulli approximate variational inference (2015), arXiv:1506.02158
12. Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: representing model uncertainty in deep learning (2015), arXiv:1506.02142

13. A. Gangwar, A. Joshi, Deepirisnet: deep iris representation with applications in iris recognition and cross-sensor iris recognition, in *2016 IEEE International Conference on Image Processing (ICIP)* (IEEE, 2016), pp. 2301–2305
14. S. Gupta, R. Girshick, P. Arbeláez, J. Malik, Learning rich features from rgb-d images for object detection and segmentation, in *European Conference on Computer Vision* (Springer, 2014), pp. 345–360
15. B. Hariharan, P. Arbeláez, R. Girshick, J. Malik, Simultaneous detection and segmentation. In *European Conference on Computer Vision* (Springer, 2014), pp. 297–312
16. H. Hofbauer, F. Alonso-Fernandez, P. Wild, J. Bigun, A. Uhl, A ground truth for iris segmentation, in *Proceedings of the 22nd International Conference on Pattern Recognition (ICPR'14)* (Stockholm, Sweden, 2014), 6pp
17. H. Hofbauer, F. Alonso-Fernandez, J. Bigun, A. Uhl, Experimental analysis regarding the influence of iris segmentation on the recognition rate. IET Biom. **5**(3), 200–211 (2016)
18. J. Huang, Y. Wang, T. Tan, J. Cui, A new iris segmentation method for recognition, in *(ICPR 2004). Proceedings of the 17th International Conference on Pattern Recognition, 2004*, vol. 3 (IEEE, 2004), pp. 554–557
19. S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015), arXiv:1502.03167
20. A. Kendall, V. Badrinarayanan, R. Cipolla, Bayesian segnet: model uncertainty in deep convolutional encoder-decoder architectures for scene understanding (2015), arXiv:1511.02680
21. W. Kong, D. Zhang, Accurate iris segmentation based on novel reflection and eyelash detection model, in *Proceedings of 2001 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2001* (IEEE, 2001), pp. 263–266
22. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
23. R.D. Labati, F. Scotti, Noisy iris segmentation with boundary regularization and reflections removal. Image Vis. Comput. **28**(2), 270–277 (2010)
24. Y. Le Cun, D. Touresky, G. Hinton, T. Sejnowski, A theoretical framework for back-propagation, in *The Connectionist Models Summer School*, vol. 1 (1988), pp. 21–28
25. Y. Le Cun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition. Proc. IEEE **86**(11), 2278–2324 (1998)
26. Y.-H. Li, M. Savvides, An automatic iris occlusion estimation method based on high-dimensional density estimation. IEEE Trans. Pattern Anal. Mach. Intell. **35**(4), 784–796 (2013)
27. N. Liu, H. Li, M. Zhang, J. Liu, Z. Sun, T. Tan, Accurate iris segmentation in non-cooperative environments using fully convolutional networks, in *2016 International Conference on Biometrics (ICB)* (IEEE, 2016), pp. 1–8
28. N. Liu, M. Zhang, H. Li, Z. Sun, T. Tan, Deepiris: learning pairwise filter bank for heterogeneous iris verification. Pattern Recogn. Lett. **82**, 154–161 (2016)
29. X. Liu, K.W. Bowyer, P.J. Flynn, Experiments with an improved iris segmentation algorithm, in *Fourth IEEE Workshop on Automatic Identification Advanced Technologies (AutoID'05)* (IEEE, 2005), pp. 118–123
30. J. Long, E. Shelhamer, T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 3431–3440
31. L. Ma, T. Tan, Y. Wang, D. Zhang, Personal identification based on iris texture analysis. IEEE Trans. Pattern Anal. Mach. Intell. **25**(12), 1519–1533 (2003)
32. H. Noh, S. Hong, B. Han, Learning deconvolution network for semantic segmentation, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1520–1528
33. D. Petrovska, A. Mayoue, Description and documentation of the biosecure software library, in *Project No IST-2002-507634-BioSecure, Deliverable* (2007)
34. A. Radman, K. Jumari, N. Zainal, Iris segmentation in visible wavelength images using circular gabor filters and optimization. Arab. J. Sci. Eng. **39**(4), 3039–3049 (2014)

35. K.B. Raja, R. Raghavendra, V.K. Vemuri, C. Busch, Smartphone based visible iris recognition using deep sparse filtering. Pattern Recogn. Lett. **57**, 33–42 (2015)
36. C. Rathgeb, A. Uhl, P. Wild, *Iris Recognition: From Segmentation to Template Security*, vol. 59, Advances in Information Security (Springer, Berlin, 2013)
37. C. Rathgeb, A. Uhl, P. Wild, H. Hofbauer, Design decisions for an iris recognition sdk, in *Handbook of Iris Recognition*, 2nd edn., Advances in Computer Vision and Pattern Recognition, ed. by K. Bowyer, M.J. Burge (Springer, Berlin, 2016)
38. C.J.V. Rijsbergen, *Information Retrieval*, 2nd edn. (Butterworth-Heinemann, Newton, 1979)
39. T. Rongnian, W. Shaojie, Improving iris segmentation performance via borders recognition, in *2011 International Conference on Intelligent Computation Technology and Automation (ICICTA)*, vol. 2 (IEEE, 2011), pp. 580–583
40. O. Ronneberger, P. Fischer, T. Brox, U-net: convolutional networks for biomedical image segmentation, in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2015), pp. 234–241
41. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014), arXiv:1409.1556
42. N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
43. C.-W. Tan, A. Kumar, Unified framework for automated iris segmentation using distantly acquired face images. IEEE Trans. Image Process. **21**(9), 4068–4079 (2012)
44. C.-W. Tan, A. Kumar, Towards online iris and periocular recognition under relaxed imaging constraints. IEEE Trans. Image Process. **22**(10), 3751–3765 (2013)
45. C.-L. Tisse, L. Martin, L. Torres, M. Robert et al., Person identification technique using human iris recognition, in *Proceedings of Vision Interface* (2002), pp. 294–299
46. P.V. Tran, A fully convolutional neural network for cardiac segmentation in short-axis mri, in *CoRR* (2016), arXiv:1604.00494
47. A. Uhl, P. Wild, Weighted adaptive hough and ellipsopolar transforms for real-time iris segmentation, in *Proceedings of the 5th IAPR/IEEE International Conference on Biometrics (ICB'12)* (New Delhi, India, 2012), pp. 1–8
48. P. Wild, H. Hofbauer, J. Ferryman, A. Uhl, Segmentation-level fusion for iris recognition, in *Proceedings of the International Conference of the Biometrics Special Interest Group (BIOSIG'15)* (Darmstadt, Germany, 2015), p. 12
49. R.P. Wildes, Iris recognition: an emerging biometric technology. Proc. IEEE **85**(9), 1348–1363 (1997)
50. Z. Zhao, K. Ajay, An accurate iris segmentation framework under relaxed imaging constraints using total variation model, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 3828–3836
51. J. Zuo, N. D. Kalka, N.A. Schmid, A robust iris segmentation procedure for unconstrained subject presentation, in *2006 Biometrics Symposium: Special Session on Research at the Biometric Consortium Conference* (IEEE, 2006), pp. 1–6

Part III
Deep Learning for Soft Biometrics

Chapter 7
Two-Stream CNNs for Gesture-Based Verification and Identification: Learning User Style

Jonathan Wu, Jiawei Chen, Prakash Ishwar and Janusz Konrad

Abstract A gesture is a short body motion that contains both static (nonrenewable) anatomical information and dynamic (renewable) behavioral information. Unlike traditional biometrics such as face, fingerprint, and iris, which cannot be easily changed, gestures can be modified if compromised. We consider two types of gestures: full-body gestures, such as a wave of the arms, and hand gestures, such as a subtle curl of the fingers and palm, as captured by a depth sensor (Kinect v1 and v2 in our case). Most prior work in this area evaluates gestures in the context of a "password," where each user has a single, chosen gesture motion. Contrary to this, we aim to learn a user's gesture "style" from a set of training gestures. This allows for user convenience since an exact user motion is not required for user recognition. To achieve the goal of learning gesture style, we use two-stream convolutional neural networks, a deep learning framework that leverages both the spatial (depth) and temporal (optical flow) information of a video sequence. First, we evaluate the generalization performance during testing of our approach against gestures of users that have not been seen during training. Then, we study the importance of dynamics by suppressing the use of dynamic information in training and testing. Finally, we assess the capacity of the aforementioned techniques to learn representations of gestures that are invariant across users (gesture recognition) or to learn representations of users that are invariant across gestures (user style in verification and identification) by visualizing the two-dimensional t-Distributed Stochastic Neighbor Embedding (t-SNE)

J. Wu · J. Chen · P. Ishwar · J. Konrad (✉)
Department of Electrical and Computer Engineering, Boston University,
8 Saint Mary's Street, Boston, MA 02215, USA
e-mail: jkonrad@bu.edu

J. Wu
e-mail: jonwu@bu.edu

J. Chen
e-mail: garychen@bu.edu

P. Ishwar
e-mail: pi@bu.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_7

of neural network features. We find that our approach outperforms state-of-the-art methods in identification and verification on two biometrics-oriented gesture datasets for full-body and in-air hand gestures.

7.1 Introduction

Biometrics are a convenient alternative to traditional forms of access control, such as passwords and pass-cards, since they rely solely on user-specific traits. Unlike alphanumeric passwords, biometrics cannot be given or told to another person, and unlike pass-cards, are always "on-hand." Perhaps the most well-known biometrics with these properties are: fingerprint, face, speech, iris, and gait.

A gesture is a short, few seconds long, body motion that contains static anatomical information and dynamic behavioral information. We consider both full-body gestures, such as a wave of the arms, and hand gestures, such as a subtle curl of the fingers and palm. For identification and verification, a user can choose a specific gesture as a "password."

In this work, rather than focusing on identifying a user performing a specific "password," we aim to identify a user across a *set* of gestures, in effect learning a user's *gesture style*. We focus on body- and hand-based gestures from depth maps acquired by Kinect sensors (v1 and v2) [13] (Fig. 7.1).

This chapter makes the following key contributions:

- development of a two-stream convolutional neural network for user identification and verification based on body and hand gestures,
- evaluation of the generalization performance of the network for gestures or users that are not seen in the training set,
- assessment of the value of dynamics for user identification and verification,
- a t-SNE-based assessment of the capacity of the studied methods to learn representations of gestures that are invariant across users (gesture recognition) or to learn representations of users that are invariant across gestures (user style in verification and identification).

We validate our approach on two biometrics-oriented datasets (BodyLogin and HandLogin), and one gesture-centric dataset (MSRAction3D).

7.2 Literature Review

Extensive literature exists for depth-based *gesture recognition* using body [9, 21, 35, 43] and hand [15, 26, 32] gestures. However, there are few works for user identification and verification based on gestures. Both body- and hand-based gesture biometrics have been investigated independently using primarily depth silhouette

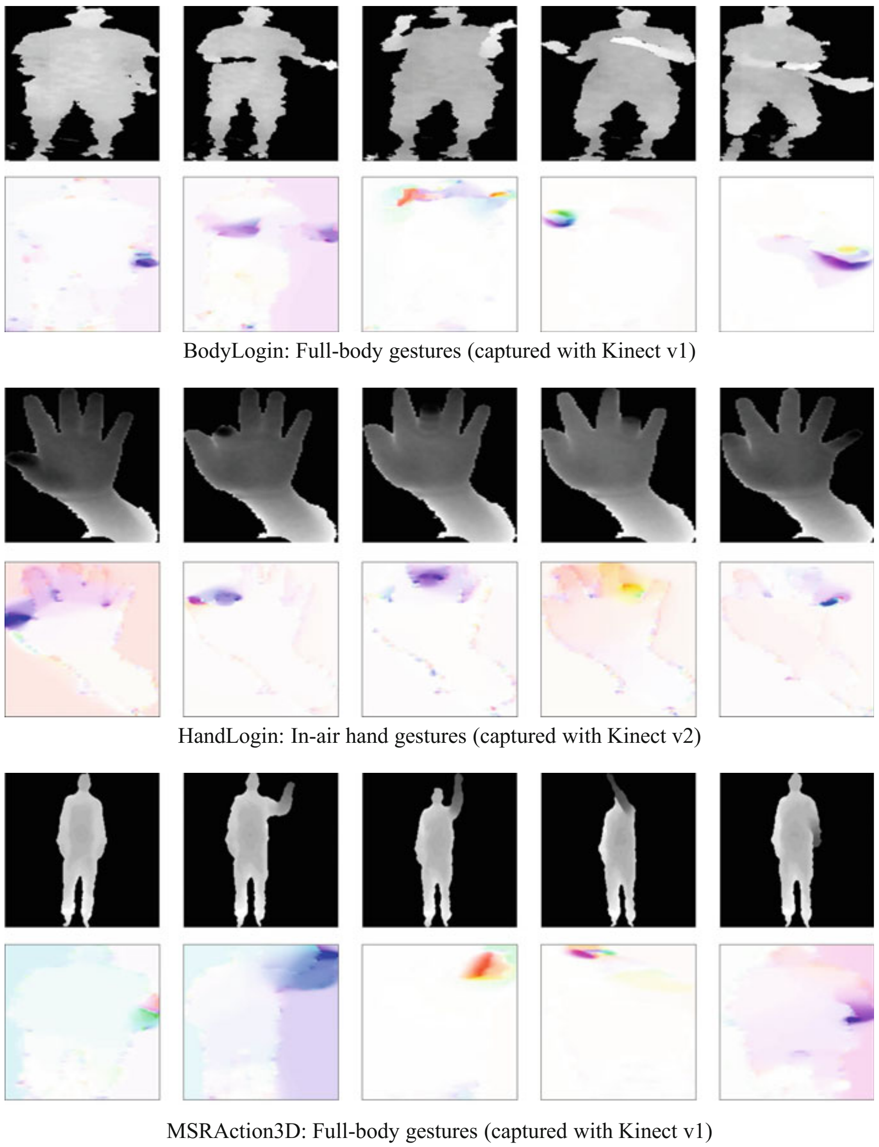

BodyLogin: Full-body gestures (captured with Kinect v1)
HandLogin: In-air hand gestures (captured with Kinect v2)
MSRAction3D: Full-body gestures (captured with Kinect v1)

Fig. 7.1 Examples of normalized depth images and corresponding colored optical flow [20] for body and hand gestures captured using various depth sensors. *Hue* indicates optical flow orientation, and *saturation* indicates magnitude

shape [39, 42] and skeletal features (pose estimates from depth maps) [1, 16, 17, 38, 41]. In [42], a temporal hierarchy of depth-based silhouette covariances from hand gestures was used to authenticate users, whereas in [1] a dynamic time warping

(DTW) algorithm applied to fingertip and palm coordinates (hand pose estimates), that were estimated from depth images, was used. Perhaps the work that is closest to the goal of this chapter is [16], where action-specific metric learning from normalized joint positions of the body was used to predict identity from a pool of known actions. We differ from that work, in that we learn user identity directly from depth images, without the need to have pose estimates of body joint positions. We use depth maps and the associated optical flow, which can be useful in cases when skeletal pose estimation is not reliable or not fully available (such as for hand poses).

In terms of methodology, we leverage the success of deep convolutional neural networks (CNNs) in various vision tasks. The goal of a CNN is to learn a large set of kernel weights optimized for a particular loss function matched to a task at hand. Within this domain, several still-image network architectures have been proposed, such as: AlexNet [14], GoogLeNet [33], and VGGNet [31]. These networks generally vary in the number of layers and the number and size of kernels. We adapt AlexNet to our task of verifying or identifying users based on gestures.

7.3 Proposed Approach

In this chapter, we analyze the biometric performance of gestures using AlexNet. AlexNet [14] is an eight-layer-deep CNN consisting of five convolutional and two fully connected layers followed by a soft-max layer. We adapt this network to gesture sequences by using a variant of the two-stream convolutional network architecture proposed in [30]. Two-stream convolutional networks, as the name implies, train two separate networks: one for spatial information, and one for temporal information. Although such networks were originally intended for RGB images, we have adapted them to handle depth maps (Fig. 7.2).

The first network is a "spatial stream" convolutional network where a stream of T input depth map frames, extracted from the input video through decimation, are mapped to a stream of T output feature vectors o_s by passing each frame, one-by-one, through the network (Fig. 7.2).

The second network is a "temporal stream" convolutional network that takes a sequence of T colored optical flow frames (corresponding to the T spatial-stream input frames) as input. Optical flow is computed from the current and next depth map image (depth map values are treated as luminance values) [20]. The computed optical flow vectors are mapped into polar coordinates and then converted to hue, based on the angle, and saturation, based on the magnitude (Fig. 7.1) [20]. Much like in the first network, this stream of T input optical flow frames is mapped to a stream of T output feature vectors o_t by passing every colored optical flow frame, one-by-one, through the temporal-stream network.

A simple convex combination of the outputs of both networks is used to yield a single output o_c which is used for performance evaluation:

$$o_c = w_s o_s + w_t o_t,$$

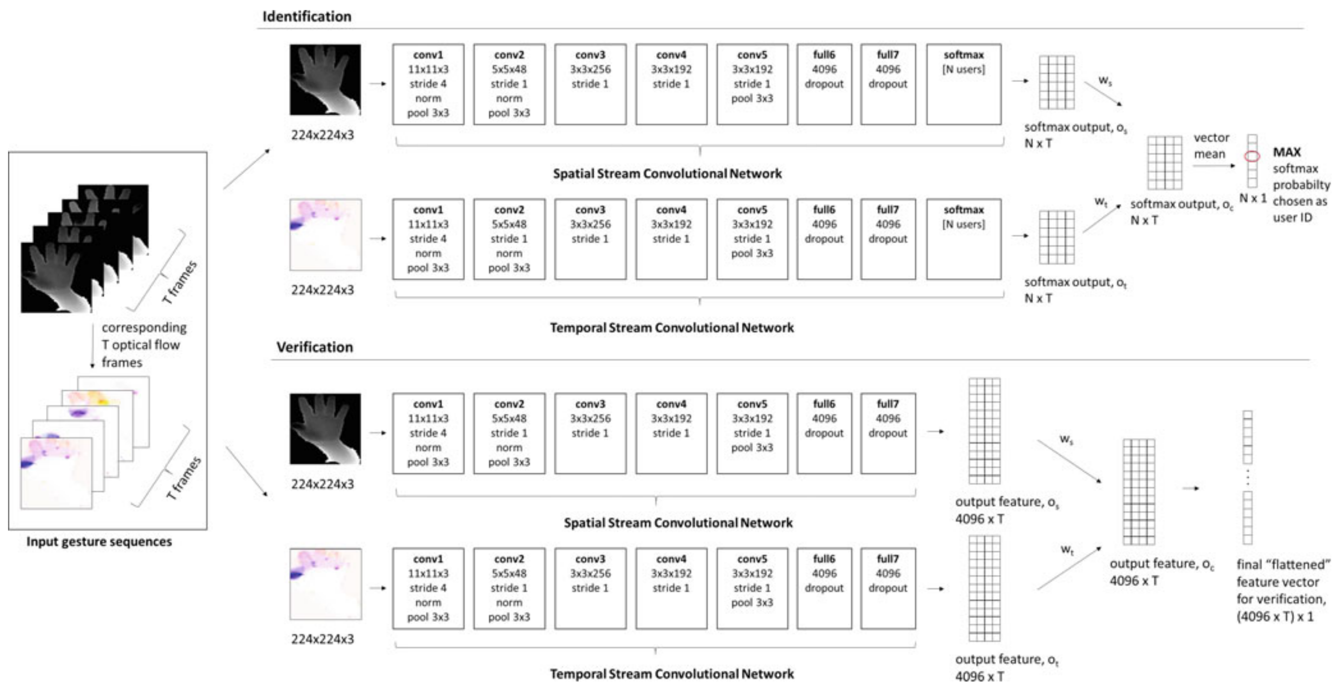

Identification
Verification
Input gesture sequences
T frames
corresponding T optical flow frames
224x224x3
conv1 11x11x3 stride 4 norm pool 3x3
conv2 5x5x48 stride 1 norm pool 3x3
conv3 3x3x256 stride 1
conv4 3x3x192 stride 1
conv5 3x3x192 stride 1 pool 3x3
full6 4096 dropout
full7 4096 dropout
softmax [N users]
Spatial Stream Convolutional Network
Temporal Stream Convolutional Network
softmax output, o_s N x T
softmax output, o_t N x T
w_s
w_t
softmax output, o_c N x T
vector mean
N x 1
MAX softmax probabilty chosen as user ID
output feature, o_s 4096 x T
output feature, o_t 4096 x T
output feature, o_c 4096 x T
final "flattened" feature vector for verification, (4096 x T) x 1

Fig. 7.2 A visualization of how we use a deep network for user identification and verification. In identification (*top*), we fully fine-tune a network, using gesture depth map frames and optical flow. In verification (*bottom*), we borrow weights from an identification network, and use the outputs of the last fully connected layer as the verification features

where $w_s \geq 0$ is the spatial-stream network weight, $w_t \geq 0$ is the temporal-stream network weight, $w_s + w_t = 1$, and o_s and o_t are the respective network outputs. When $w_s = 1, w_t = 0$, only information from the spatial-stream network is used, and when $w_s = 0, w_t = 1$, only information from the temporal-stream network is used. We will report results for various combinations of (w_s, w_t) weights.

7.3.1 CNNs for Identification and Verification

Identification: The use of this network for *closed-set identification*, i.e., given a gesture, *identify* a user from a set of known users, is straightforward. During training (see Sect. 7.3.2), gesture sequences are broken up into single frames to be used standalone. During testing, we take the mean of the soft-max probability outputs o_c across T frames (Fig. 7.2). Recall that o_c is a weighted combination of the soft-max probabilities for an input across two networks. This yields a single soft-max probability vector of length N (given N users to identify), and the component with the largest probability identifies the user. Although not the main focus of this chapter, gesture recognition uses the same structure where N is the number of gestures rather than the number of users to identity.
Verification: In *verification*[1] (given a gesture, is a user who (s)he claims to be?), we propose using the output features from the "full7" layer of a network trained for identification (Fig. 7.2). This avoids having to train a separate verification network for each user which is very expensive computationally. In addition, there are simply not enough training samples for each positive class represented by an authentic user to fully train a network. In this approach, for T frames that are uniformly sampled from a gesture sequence, two features of dimension $4096 \times T$ (the length of the last fully connected layer) are extracted yielding o_s and o_t, whose linear combination gives o_c. Since there is no built-in classification in this approach, we use these features as the input to a two-class classification algorithm for verification (we use a 1-nearest-neighbor classifier). The intuition behind this idea is that, given enough users to identify, the network will naturally learn a user-separating feature space which can be leveraged for verification.

We discuss the parameters and training of all the elements of our networks in the next section.

7.3.2 Network Implementation Details

Typically, there are not enough training samples in gesture datasets to train all the weights of a deep convolutional network from scratch. Therefore, we follow the common practice to "pre-train" the network [5, 12] using weights from another

[1] Verification is also called authentication.

network with sufficient data and then fine-tune those weights for new data. In our case, we use the dataset from ImageNet [27] to train a network with a soft-max loss function, that classifies RGB images into 1000 classes, to initialize the weights in our 5 convolutional layers (conv1 to conv5). Although our modality is different, as we use depth images and colored optical flow (instead of RGB), initializing with ImageNet weights is still effective. Our fully connected layers are trained from scratch, with weights initialized to be zero-mean Gaussian with a small standard deviation of 0.001. In all our networks, we use a batch size of 256 images. For the spatial-stream network, we start with a learning rate of 0.003, decreasing this rate by one-tenth every 3,000 iterations until a total of 12,000 iterations are completed. For the temporal-stream network, we start with a learning rate of 0.001, decreasing this rate by one-tenth every 1,000 iterations until a total of 6,000 iterations are completed. The dropout value is set to 0.5 in the fully connected layers of both networks.

We implement, in entirety, all our networks using Caffe [11, 37] on a single Titan Z GPU.

7.4 Gesture Datasets

We evaluate our method on 3 publicly available datasets. Two of these datasets were designed for user verification and identification (collected with the intention of maximizing the number of users). The third one was designed for gesture recognition (collected with the intention of maximizing the number of gesture/action types).

Notably, datasets for gesture recognition are typically gesture-centric meaning that they have a high number of gestures per user (many gestures to classify, few users performing them) whereas studying authentication requires the opposite, namely a user-centric dataset which has a high number of users per gesture. This issue is highlighted in Table 7.1, where we compare several gesture recognition datasets. Clearly, many of these datasets contain less than 20 users. In cases where there are more than 20 users, the data has been collected in such a way that there is either not enough users performing each gesture (CMU Mocap) or the data contains dataset bias due to gestures being performed continuously standing in-place (MSRC-12). By standing in-place, each user's lower body posture does not significantly change which can cause dataset bias. In gesture recognition, this is typically not an issue, as the same user will almost never be seen in *both* training and testing. However, for cases of user recognition, this causes a significant issue, as the same user is almost always seen in *both* training and testing.

Gesture-Based User Identification Datasets

HandLogin [8, 42] is a dataset containing in-air hand gesture sequences of 21 users, each performing 4 different gestures that are recorded by a Kinect v2 sensor. All 4 hand gestures are performed with one's right hand starting from a "rest" position: the hand extended downwards on top of a ceiling-facing Kinect sensor, with fingers comfortably spread apart. This placement of the sensor avoids the notorious "gorilla

Table 7.1 A comparison of mostly *body* gesture recognition datasets using either depth or Mocap (motion capture) sensors. We have highlighted, user-centric datasets (ideal for identification and authentication) in bold. $^{\diamond}$In CMU Mocap, all users do not perform all gestures (some gesture types have only a single user performing it). ‡In MSRC-12, gestures are performed continuously in a long sequence, one after another causing inherent biases

Dataset	# of users	# of gestures	Data type
CMU Mocap [3]	>100	$109^{\diamond}$	Mocap
HDM05 [22]	5	>70	Mocap
MSRAction3D [18]	10	20	Kinect v1 Depth
HumanEva I/II [29]	4/2	6/1	RGB + Mocap
MSRC-12 [7]	30	$12^{\ddagger}$	Kinect v1 (Skeletons only)
MSRGesture3D [36]	10	12	Kinect v1 Depth
MSRDailyActivity3D [35]	10	16	Kinect v1 Depth
Berkeley MHAD [23]	12	11	Multimodal (Depth + Mocap)
BodyLogin [4]	40	5	Kinect v1 Depth
Handlogin [8]	21	4	Kinect v2 Depth

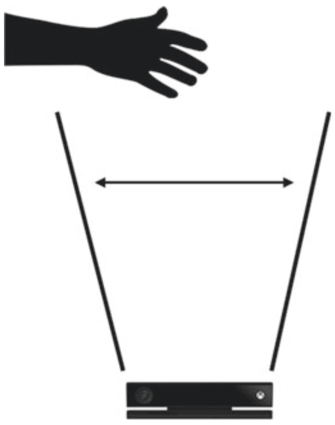


Fig. 7.3 Visualization of HandLogin camera setup. Kinect v2 camera points towards the ceiling

arm" issue, where users would need to maintain their hand in a vertical front-to-parallel position instead of using a more comfortable horizontal down-to-parallel position (see Fig. 7.3). The orientation of our sensor was designed to mimic an authentication terminal, where typically only a single user is visible. The following gestures are performed (Fig. 7.4): compass (move open hand in multiple directions), piano (move fingers as if playing piano), push (move open hand towards and away from the sensor), and flipping fist (twist and curl hand into a fist). Each user performed 10 samples of each gesture.

BodyLogin [2, 39–41] is a full-body multi-view dataset containing gesture sequences of 40 users performing 5 different gestures that are recorded by Kinect v1

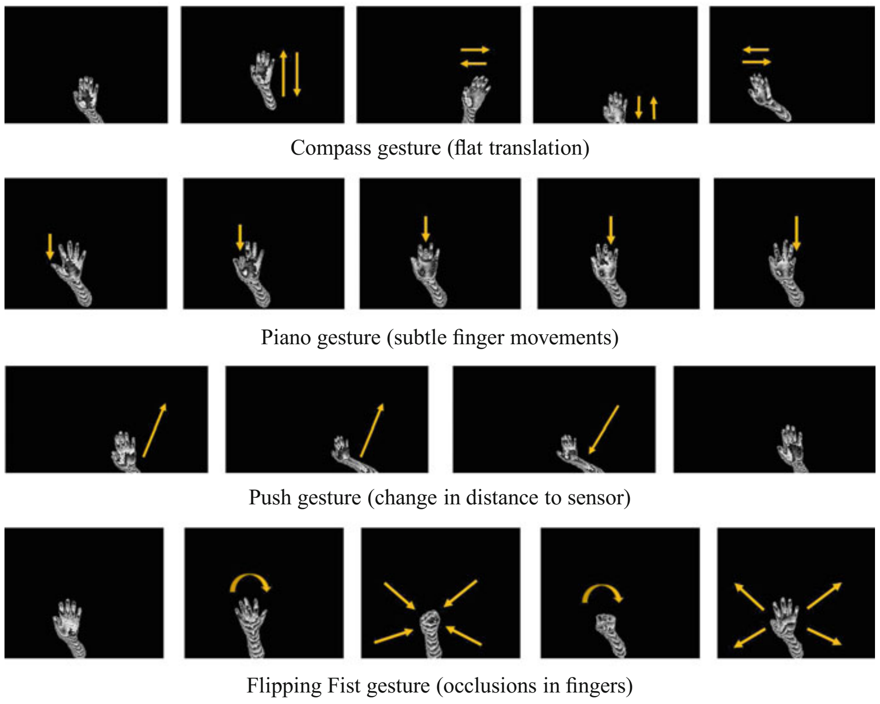
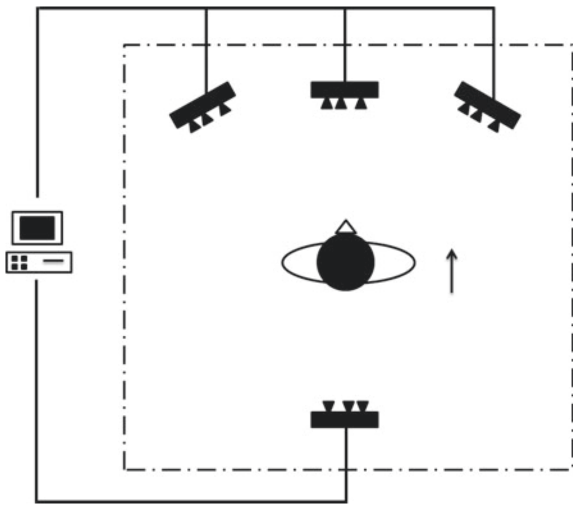

Compass gesture (flat translation)
Piano gesture (subtle finger movements)
Push gesture (change in distance to sensor)
Flipping Fist gesture (occlusions in fingers)

Fig. 7.4 The 4 gestures used in HandLogin for user identification. For visualization, images have been cropped, and only show the lower 4-bits of the 16-bit depth image

Fig. 7.5 BodyLogin camera setup with four Kinects. Three Kinects (*left*, *center*, and *right*) were placed in front of the user, and one was placed behind the user (*back*)



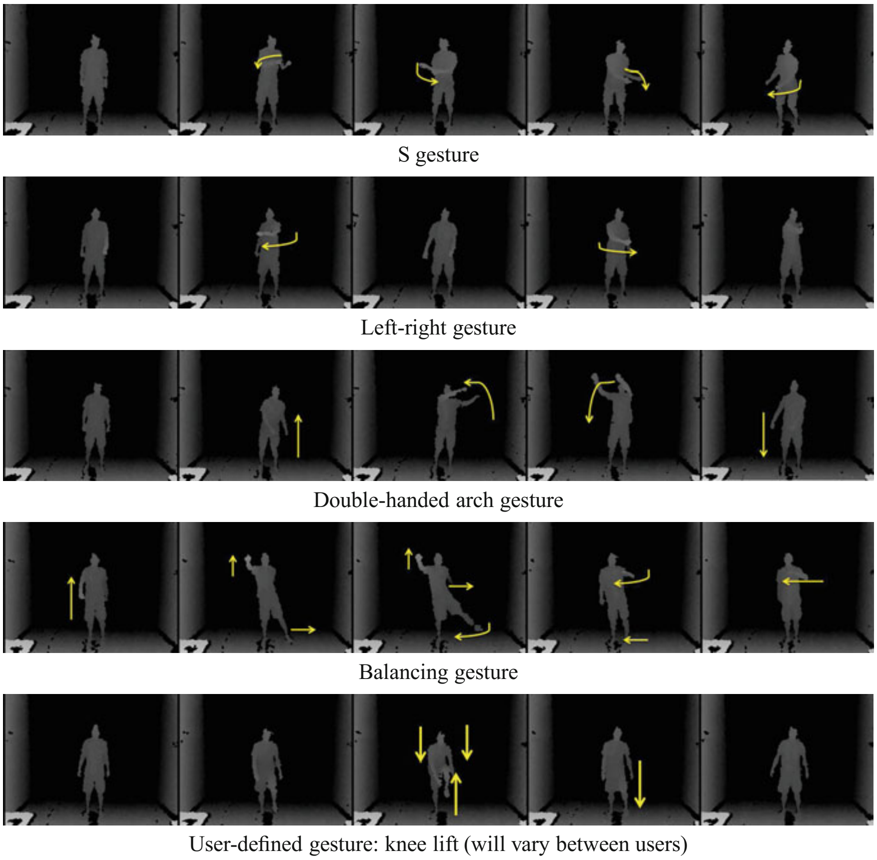

S gesture
Left-right gesture
Double-handed arch gesture
Balancing gesture
User-defined gesture: knee lift (will vary between users)

Fig. 7.6 The 5 gestures used in BodyLogin for user identification

sensors (see Fig. 7.5). Four of these gestures are predefined: S gesture (user draws an "S" shape with both arms), left-right (user reaches right shoulder with left hand, and then left shoulder with right hand), double-handed arch (user moves both arms in an upwards arch), and balancing (user performs a complex balancing gesture involving arms and legs). The fifth gesture is created by the user (user defined). These gestures are visualized in Fig. 7.6. Each user performed each gesture about 20 times under varying degradations, such as carrying a bag, wearing a coat, passage of time, and also under spoof attacks. In this study, we train and test with samples across all degradations, and only from the *center* camera viewpoint. For download details of the *HandLogin* and *BodyLogin* datasets see Sect. 7.6.

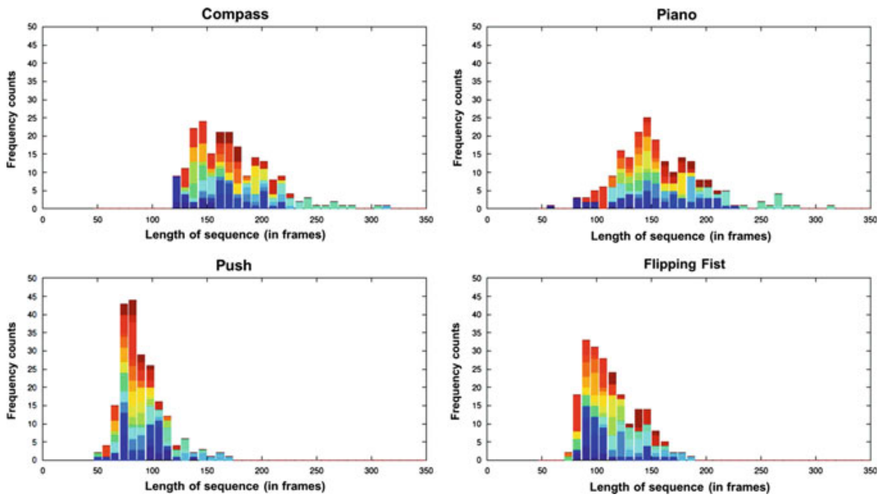


Fig. 7.7 Stacked histograms representing the lengths of the 4 gestures in HandLogin. *Color* denotes sample frequency counts from a specific user. Gestures were collected at 30 frames per second

In both of these identification-focused datasets, the duration of a single gesture sample is only a few seconds. User-specific durations for fixed gesture types are shown in Figs. 7.7 and 7.8 as stacked histograms, where color denotes sample frequency counts from a specific user. For the most part, these datasets consider gestures that are intentionally short. This is useful as performing a gesture that is too long becomes harder to remember and repeat. Further, having to perform a long gesture can become too prohibitive and too inconvenient over other alternative forms of access control. It is important to note that gestures that are of longer duration do not necessarily yield better performance. There can be scenarios where a gesture is too hard to remember, or too hard to replicate consistently, which can result in poor biometric performance relative to a shorter and simpler gesture.

Gesture Recognition Dataset

MSRAction3D [18, 35] is a full-body, single-view dataset containing motion sequences of 10 users, performing 20 different actions in front of a Kinect v1 sensor. Each subject performs each action 2 or 3 times, with a total of 567 depth map sequences. Actions in this dataset are quite varied, for example: arm waves, hammer motions, catches, punches, symbol drawings, kicks, tennis swings, golf swings, and jogging. Although in [18], the actions are split into 3 non-overlapping subsets (each containing 1/3 of actions) for evaluation, we instead evaluate all the actions at once in all our experiments, which is a more difficult scenario.

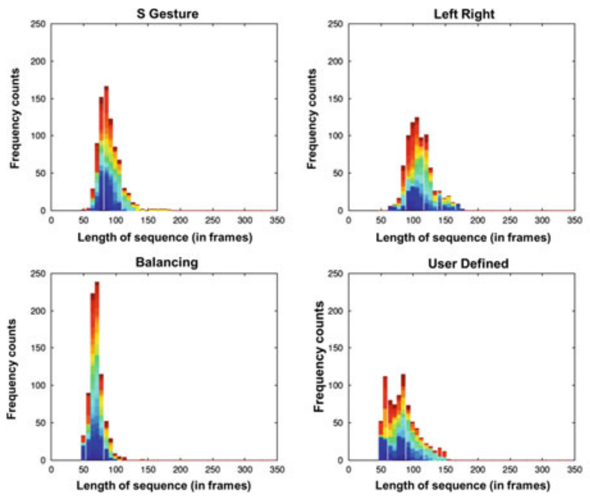


Fig. 7.8 Stacked histograms representing the lengths of the 5 gestures in BodyLogin. *Color* denotes sample frequency counts from a specific user. Gestures were collected at 30 frames per second

Dataset Preprocessing

First, background subtraction (background frames are given) was applied to all depth frames in all datasets. Then, the background-subtracted frames were scaled to 0–1 range and resized using bicubic interpolation to 224×224 pixels as shown in Fig. 7.1.

7.5 Performance Evaluation

We evaluate performance for two access control scenarios [10]: closed-set identification and verification.

In *closed-set identification*, given a query gesture sequence, an identity is predicted from a pool of known users. The performance measure we use for identification is the correct classification error (CCE), which is the rate at which users are incorrectly identified.

In *verification*, given a query gesture sequence and claimed identity, the claim is either verified or rejected. If the query is sufficiently close in distance to a known, enrolled gesture sequence of the claimed identity, it will be accepted as that user; otherwise, it will be rejected. An error in verification results from either a false acceptance or a false rejection. The false acceptance rate (FAR) is the rate at which *unauthorized* users are accepted and is a measure of security. The false rejection rate (FRR) is the rate at which *authorized* users are denied access and is a measure of convenience. There exists a trade-off between FAR and FRR which is controlled by a threshold on acceptance distance (between the query and closest enrolled gesture).

A popular metric that captures this trade-off with a single scalar is the equal error rate (EER) which is the FAR (or FRR) for the threshold when FAR and FRR are equal.

In our verification experiments, we use the ℓ_2 distance between the features of gesture sequences (flattened vectors of length $4096 \times T$, $T = 50$). If the distance, d_{NN}, between a query sample $\mathbf{Q}$ and its nearest-neighbor enrolled sample of the claimed identity is below a threshold, it is accepted; otherwise, it is rejected. In this chapter, we report the EER for our verification experiments.

In detail, let $\mathscr{A}_i$ denote the set of gesture samples from a single authorized user i, and let $\mathscr{U}_i$ denote the set of gesture samples that do not come from authorized user i. Then, for a given threshold value θ, the FAR and FRR are calculated by:

$$FRR(\mathscr{A}_i, \theta) = \frac{\sum_{\mathbf{Q}\in\mathscr{A}_i} \mathbf{1}(d_{NN}(\mathbf{Q}, \mathscr{A}_i\backslash\mathbf{Q}) \geq \theta)}{|\mathscr{A}_i|}$$
$$FAR(\mathscr{A}_i, \mathscr{U}_i, \theta) = \frac{\sum_{\mathbf{Q}\in\mathscr{U}_i} \mathbf{1}(d_{NN}(\mathbf{Q}, \mathscr{A}_i) < \theta)}{|\mathscr{U}_i|}$$

where the indicator function $\mathbf{1}(condition)$ equals 1 if the '$condition$' is true and equals 0 otherwise.

EER for the pair $(\mathscr{A}_i, \mathscr{U}_i)$ is found by first computing the FAR-FRR pairs for different thresholds θ. Then, the EER is determined as the location on the boundary of the convex hull of the FAR-FRR pairs where FAR equals FRR. In practice, this EER may not lie directly on an FAR-FRR pair that corresponds to any decision threshold. In general, the EER point represents the FAR-FRR performance of a randomized decision rule that chooses, with some probability, between two decision rules having different thresholds [28]. This computation with a fixed threshold θ can be repeated and averaged across all authorized users who each have his/her own unique set $(\mathscr{A}_i, \mathscr{U}_i)$.

7.6 Experimental Results

In all our experiments, we benchmark against reimplemented depth silhouette covariance features as proposed in [42]. This method is not based on convolutional neural networks. For convenience, we have combined the *HandLogin* and *BodyLogin* datasets into a single download available on our website [4].

User Identification

We attempt to identify a user across a whole pool of possible gestures. We test performance both when a gesture has been seen by the system and also when it has not. The latter case evaluates how well our learned model *generalizes* to gestures that have not been part of the training set. If it performs well, our model would have, in effect, learned a specific "style" with which a user performs gestures, not just the specific gestures a user performs.

Table 7.2 User identification results for HandLogin and BodyLogin

Dataset	Scenario	**User Identification CCE (%)**					
		Weighted Convnets (w_s, w_t)					Baseline
	Training/Testing Gestures	⟵ Spatial			Temporal ⟶		Wu [42]
		$(1, 0)$	$(\frac{2}{3}, \frac{1}{3})$	$(\frac{1}{2}, \frac{1}{2})$	$(\frac{1}{3}, \frac{2}{3})$	$(0, 1)$	
HandLogin (21 users, 4 gestures)	1. All/All	0.24	0.24	**0.24**	0.71	4.05	6.43
	2. All but Compass/Compass	**2.38**	2.86	4.76	8.57	36.19	82.38
	3. All but Piano/Piano	1.91	**0.48**	1.43	1.91	12.86	68.10
	4. All but Push/Push	**44.29**	49.05	54.29	67.62	77.14	79.52
	5. All but Fist/Fist	16.67	**15.71**	17.14	20.00	31.43	72.38
BodyLogin (40 users, 5 gestures)	1. All/All	0.05	0.05	0.05	**0.05**	5.01	1.15
	2. All but S/S	**0.75**	1.00	1.25	1.75	16.75	75.75
	3. All but Left-Right/Left-Right	**0.88**	1.25	1.50	1.88	11.50	80.88
	4. All but 2-Handed Arch/2-Handed Arch	0.13	0.13	**0.13**	0.38	6.25	74.50
	5. All but Balancing/Balancing	**9.26**	10.01	13.27	19.52	45.06	77.97
	6. All but User Defined/User Defined	**5.28**	5.53	6.16	8.54	22.49	71.61

Table 7.3 User identification on MSRAction3D. [16] performs user identification based on skeletal pose estimates derived from depth maps

Dataset	**User Identification CCE (%)**				
	Weighted Convnets (w_s, w_t)			Baselines	
	⟵ Spatial		Temporal ⟶	Wu [42]	[16]
	(1, 0)	$(\frac{1}{2}, \frac{1}{2})$	(0, 1)		
MSR	0.0	**0.0**	0.53	13.6	7.0

Results for both the BodyLogin and Handlogin datasets are shown in Table 7.2. The first row of this table ("All/All") refers to a scenario when the network has been trained with samples from all gestures. In this row, we split the dataset into one half for training and the other half for testing, where each half contains samples from all gestures. The remaining rows in the table are for scenarios when the network has been trained on some gestures while tested on a different unseen gesture. For example, for "All but Fist/Fist" the network has been trained on "Compass," "Piano," and "Push" but tested on "Fist." In Table 7.3, we report results for user identification on the MSRAction3D dataset. Here, we train only on one sample of each action, and test on the remaining 1–2 samples. This is the same as the row ("All/All") in Table 7.2, where we train with samples from all gestures. In addition to our silhouette covariance benchmark from [42], we also compare to the reported user identification results from [16], which uses skeletal joint estimates and a distance metric based on skeletal coordinates, to determine user identity.

Suppression of Dynamics in User Identification

In order to understand the impact of dynamics in our deep network representation, we studied the effect of "removing" it. Although a similar study was done in [40], that was based on skeletal pose estimates. Our study is based on depth maps. We consider *both* the input to the temporal-stream network, as well as the input to the spatial-stream network as containing full dynamic information. To suppress the impact of dynamics, we remove the temporal network completely, and use only the first 3 depth map frames (out of typically hundreds of frames, thus spanning the time duration of less than a tenth of a second) as input to the spatial-stream network. In Table 7.4, we show the empirical performance of dynamics suppression for our two-stream approach as well as for the approach in [42] which we have reimplemented for this experiment.

User Verification

Here, we attempt to verify a user's query gesture and claimed identity against a pool of known gestures (all gestures of the claimed identity). As it is impractical to train a deep network for each user, we instead train an identification network first and use it as a *feature extractor* for verification (see Sect. 7.3). In our experiments, we "leave-out" one-fourth of the user pool for testing, and train an identification network (for feature extraction) on the remaining three-fourths. For BodyLogin, this is leave-10-persons-out and for HandLogin this is leave-5-persons-out cross-validation. In the benchmark verification method, we use covariance features from the test samples. In Table 7.5, we report these results averaged across 4 "leave-out" folds for verification on Bodylogin and HandLogin datasets.

Gesture Recognition

Here, we attempt to recognize the gesture type performed across a pool of users. While in user identification we are trying to learn the user identity irrespective of which gestures the user performs, in gesture recognition we are trying to learn the gesture irrespective of the users who perform them. Similar to how we "leave-out" gestures in our user identification experiments, we "leave-out" users in our gesture recognition experiments. Specifically, we "leave-out" half of the user pool for testing, and train a gesture recognition network on the remaining half. For MSRAction3D, we employ the cross-validation approach of leaving 5 persons out as done in [24], and in BodyLogin[2] and Handlogin, we perform leave-20-persons-out, and leave-10-persons-out (half of each dataset population), respectively. We report results for gesture recognition in Table 7.6.

Table 7.4 Results for the suppression of dynamics in user identification: only first 3 frames of each depth map sequence are used for training and testing, and the temporal stream is disabled ($w_s = 1, w_t = 0$)

Dataset	Scenario	**User Identification CCE (%)**	
	Data used	Spatial	Wu [42]
HandLogin	All frames	**0.24**	6.43
	No dynamics	**1.90**	9.29
BodyLogin	All frames	**0.05**	1.15
	No dynamics	**1.00**	32.60

Table 7.5 User verification results for HandLogin and BodyLogin

Dataset	Scenario	**User Verification EER (%)**					
		Weighted Convnets (w_s, w_t)					Baseline
	Users	$\longleftarrow$ Spatial			Temporal $\longrightarrow$		Wu [42]
		(1, 0)	$(\frac{2}{3}, \frac{1}{3})$	$(\frac{1}{2}, \frac{1}{2})$	$(\frac{1}{3}, \frac{2}{3})$	(0, 1)	
HandLogin	Leave-5-persons-out	2.52	**2.20**	2.71	4.09	6.50	11.45
BodyLogin	Leave-10-persons-out	2.76	2.45	**1.99**	3.07	8.29	3.46

[2]Of the 5 gesture classes in BodyLogin, 4 gesture classes are shared across users, and 1 is not, being user defined. This means that in leave-persons-out gesture recognition, the fifth gesture class will not have samples of its gesture type in training. As a result, the fifth gesture class is expected to act as a "reject"/"not gestures 1-4" category for gesture recognition.

Table 7.6 Gesture recognition results. For each dataset, we perform leave-($N/2$)-persons-out cross-validation, where N is equal to the total number of users in the dataset

Dataset	**Gesture Recognition CCE (%)**			
	Weighted Convnets (w_s, w_t)			Baseline
	$\longleftarrow$ Spatial		Temporal $\longrightarrow$	Wu [42]
	(1, 0)	$(\frac{1}{2}, \frac{1}{2})$	(0, 1)	
HandLogin	15.00	6.82	10.91	**0.91**
BodyLogin	21.10	**15.09**	20.35	15.44
MSRAction3D	44.36	36.00	40.36	**25.45**

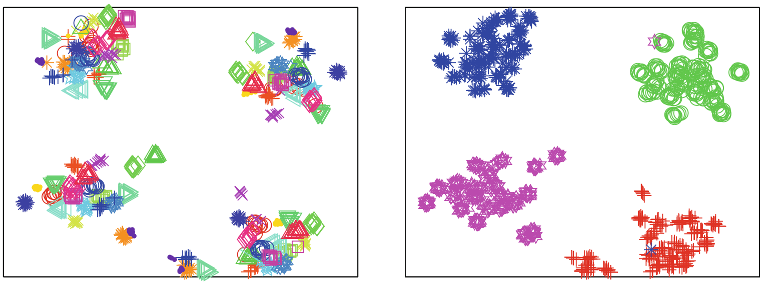
7.7 Discussion

The above results demonstrate a significant decrease in error when using deep networks compared to benchmark methods in user identification (all 3 datasets) and verification (HandLogin and BodyLogin).[3] This decrease is most striking in identification, when we test gestures that have not been used in training the network. In stark contrast to the CNN features proposed in our work, the covariance features proposed in [42] do not generalize well *across* gestures, i.e., when gestures that are not part of the training set appear in the test set. This can be seen most clearly by examining the CCE values for the "Compass" gesture in Table 7.2. The CCE for covariance features is as high as 82.38% while it is only 2.38% for our CNN features.

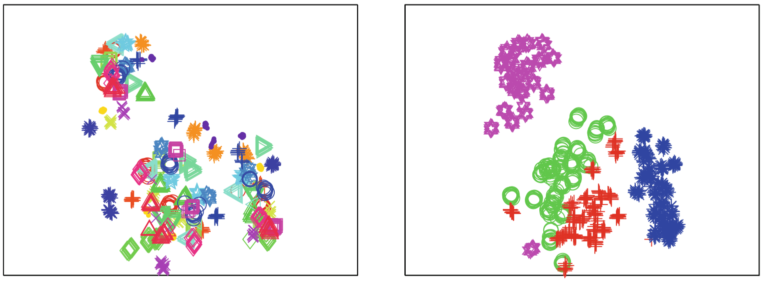
This cross-gesture generalization capacity of CNNs is also observed in the t-SNE embeddings [34] of the "full7" layer outputs for Handlogin (Fig. 7.9), BodyLogin (Fig. 7.10), and MSRAction3D (Fig. 7.11) datasets. Part (a) of each figure shows the feature embedding for our baseline, which favors clustering by gesture type. Parts (b), (c), and (d) show the feature embeddings for our convolutional networks. In part (b), the pre-trained embedding from ImageNet tends to favor clustering points by gesture type. After fine-tuning for identification in part (c), we see clustering by user identity. This reveals that it is very beneficial to fine-tune our networks from the pre-trained weights in order to cluster by user. Fine-tuning for gesture recognition, shown in part (d), causes even more compact clustering by gesture type than in part (b). Note that in the t-SNE plots of the "full7" layer outputs after fine-tuning for identification (part (c)) users tend to cluster together whereas gesture types are mixed within each cluster. However, in the corresponding t-SNE plots of the covariance features (part (a)), gesture types tend to cluster together with users being mixed within each cluster.

There are cases where our network does not generalize well across gestures, e.g., the "Push" gesture (Table 7.2). We posit that this lower performance occurs because the trained gestures are significantly different in form and dynamics from the other gestures. The "Push" gesture contains variations in *scale* whereas the other gestures do not. The "Fist" gesture contains motion that completely occludes the shape of the

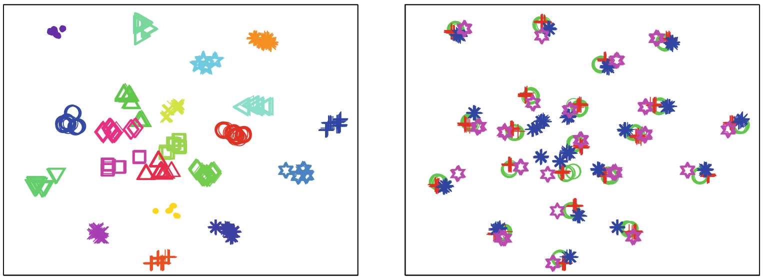
[3]Due to the general lack of per-user samples in MSRAction3D (as it is a gesture-centric dataset), we do not report results for verification, and leave-gesture-out experiments for identification.



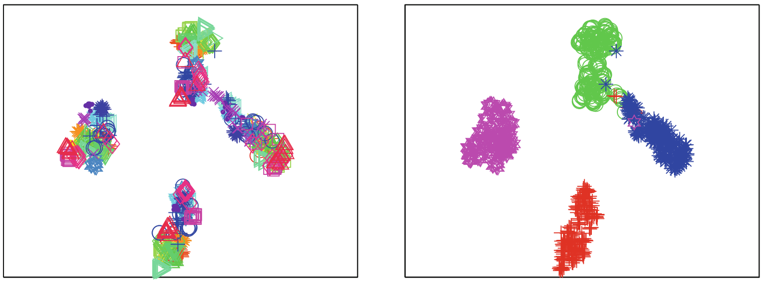
(a) HandLogin silhouette covariance features [42]



(b) HandLogin pre-trained “full7” features (no fine tuning)

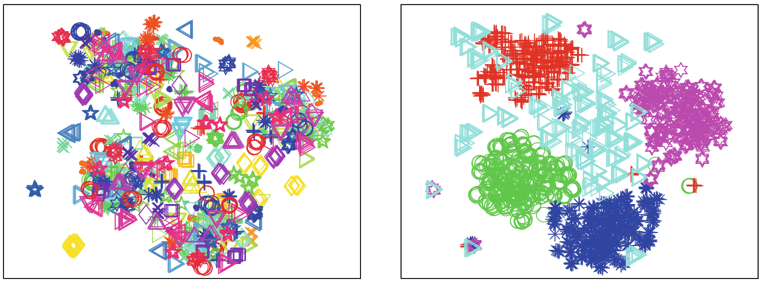


(c) HandLogin *user identification* fine-tuned “full7” features

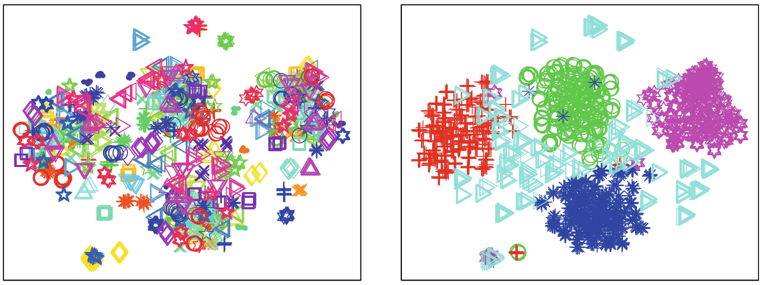


(d) HandLogin *gesture recognition* fine-tuned “full7” features

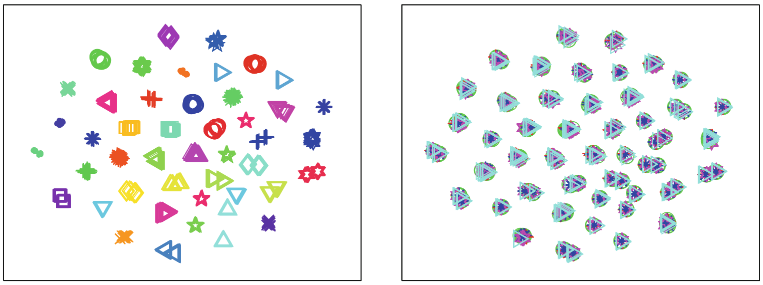
Fig. 7.9 2-D t-SNE embeddings of features for the HandLogin dataset. *Left-column* plots are color-coded by user, whereas those in the *right column* are color-coded by gesture type. A single marker represents a single gesture sequence. These figures show the t-SNE embeddings of the last fully connected layer’s output from our convolutional networks (before and after fine-tuning), and those from our baseline, silhouette covariance features



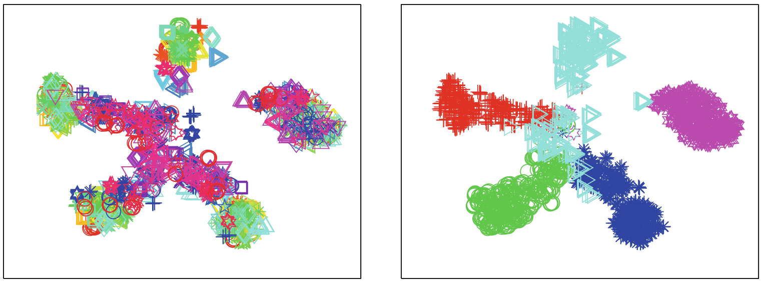
(a) BodyLogin silhouette covariance features [42]



(b) BodyLogin pre-trained “full7” features (no fine tuning)

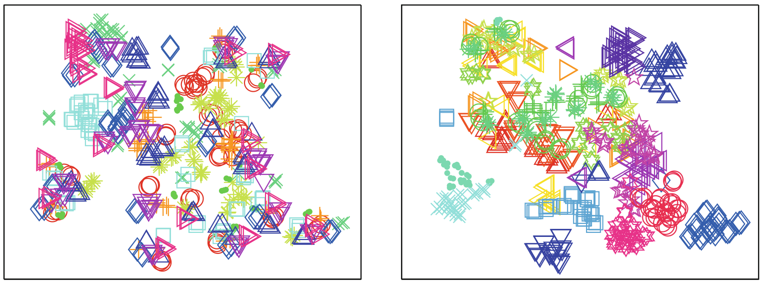


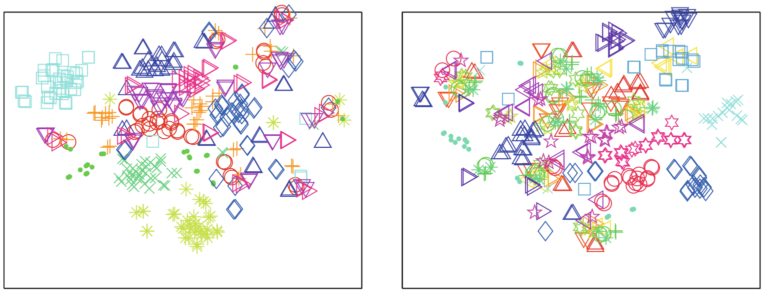
(c) BodyLogin *user identification* fine-tuned “full7” features

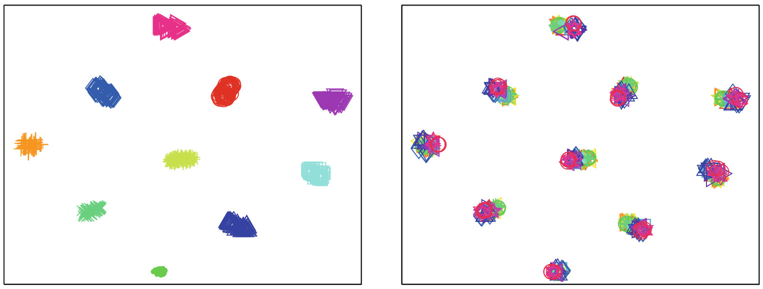


(d) BodyLogin *gesture recognition* fine-tuned “full7” features

Fig. 7.10 2-D t-SNE embeddings of features for the BodyLogin dataset. For additional information, please see Fig. 7.9. The *cyan marker* denotes user-defined gestures where any motion is allowed; it is not expected to cluster tightly


(a) MSRAction3D silhouette covariance features [42]


(b) MSRAction3D pre-trained “full7” features (no fine tuning)


(c) MSRAction3D *user identification* fine-tuned “full7” features

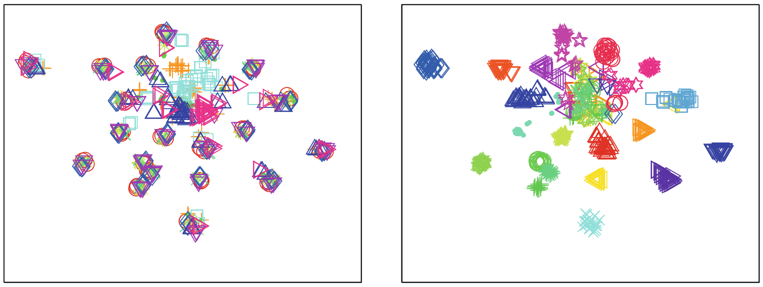

(d) MSRAction3D *gesture recognition* fine-tuned “full7” features

Fig. 7.11 2-D t-SNE embeddings of features for the MSRAction3D dataset. For additional information, please see Fig. 7.9

hand, which is not present in the other gestures. The "Balancing" gesture includes leg movements, not so for other gestures. For the most part, this type of result is to be expected. It will always be difficult to generalize to a completely unknown gesture that has little-to-no shared components with training gestures.

For identification on MSRAction3D, we get 0% classification error. Although seemingly surprising, this result might be attributed to the dataset collection procedure. In MSRAction3D, gesture samples from a user are extracted by partitioning one long continuous video into multiple sample parts. While not an issue for gesture recognition (as the same user will never be in *both* training and test sets due to "leave-persons-out" testing), this can result in biases for user recognition. This bias stems from almost identical, partially shared body postures across samples, which the deep network learns very well. The aforementioned issue is avoided in BodyLogin and HandLogin, as there is a "reset" procedure between samples – samples are *not* recorded in one long continuous sequence (users leave and re-enter the room between samples).

For verification, the differences are far less dramatic, but CNN features still yield a decent decrease in EER (Table 7.5). In both scenarios, the smaller the value, the better the performance (we want small EER and CCE).

Across all our results, the temporal stream is complementary to the spatial stream for user identification, verification, and even gesture recognition. That is, having a temporal-stream weight $w_t \neq 0$, will not degrade performance. The only exception to this is when *information* is not seen in the training phase such as in leave-gesture-out results for user identification in Table 7.2. The reduced performance due to the inclusion of the temporal stream is not entirely surprising, as there are body/hand motions in testing that have not been seen in training (unseen optical flow vectors). As a result, this ends up generalizing poorly, whereas the static poses from the spatial network still fare quite well. Across all experimental results, a simplistic weighted average of $(\frac{1}{2}, \frac{1}{2})$ is perhaps the best option.

Our experiments involving dynamics suppression in user identification (Table 7.4) confirm that motion plays a crucial role; it can reduce the mis-identification rate from 1 error in 100 attempts to 1 error in 2,000 attempts (for BodyLogin). This conclusion is consistent across our proposed method and the benchmark we evaluate.

In gesture recognition, our deep learning approach slightly outperforms the non-CNN approach on BodyLogin, but is outperformed on the other datasets. We speculate that this is due to the size of the dataset. Notably, BodyLogin is our largest dataset with the most samples ($\approx$4,000 gesture sequences, $\approx$150 frames each), and can beat our baseline. This is larger than both HandLogin ($\approx$840 gesture sequences, $\approx$150 frames each) and MSRAction3D ($\approx$600 gesture sequences, $\approx$35 frames each) combined, both of which underperform in gesture recognition. As the CNN approach outperforms the baseline in all other experiments, this perhaps suggests that with fewer samples it is easier to discriminate between users, than it is to discriminate between gestures. Overall, we believe that on larger datasets such as BodyLogin, deep learning will likely outperform the baseline.

7.8 Conclusions

This is the first work to investigate the use of two-stream convolutional networks for learning user-specific gesture "styles". Most prior works assume a single gesture password per user and perform poorly when gesture types that are not encountered in the training set appear during testing. The proposed CNN-based features are able to effectively generalize across multiple types of gestures performed by the same user by implicitly learning a representation that depends only on the intrinsic "style" of each user as opposed to the specific gesture as we demonstrated across multiple datasets.

A key practical outcome of this approach is that for verification and identification there is no need to retrain a CNN as long as users do not use dramatically different gestures. With some degradation in performance, a similar new gesture can still be used for convenience.

A direction for future work could explore recent advances in two-stream convolutional networks that involve fusing the spatial and temporal streams [6, 19, 25]. One way this can be done, is by fusing the outputs of both the spatial and temporal-streams' convolutional layers with a simple operation such as sum, multiply, or max. This method has shown promising improvements in the action recognition literature, and may lead to improvements in user identification/verification, where we are using gestures as a biometric. A key benefit of such an approach, would be that explicit weighting of the temporal and spatial-stream networks would no longer be needed; rather the contributions of each network would be learned automatically in a complete end-to-end architecture.

Additional information and resources for this work are available at our website [4].

References

1. M. Aumi, S. Kratz, Airauth: evaluating in-air hand gestures for authentication, in *Proceedings of the 16th International Conference on Human-Computer Interaction with Mobile Devices & Services* (ACM, 2014), pp. 309–318
2. BodyLogin, http://vip.bu.edu/projects/hcis/body-login
3. CMU Motion Capture Database, http://mocap.cs.cmu.edu
4. DeepLogin, http://vip.bu.edu/projects/hcis/deep-login
5. J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, T. Darrell, Decaf: a deep convolutional activation feature for generic visual recognition, in *Proceedings of The 31st International Conference on Machine Learning* (2014), pp. 647–655
6. C. Feichtenhofer, A. Pinz, A. Zisserman, Convolutional two-stream network fusion for video action recognition, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 1933–1941
7. S. Fothergill, H. Mentis, P. Kohli, S. Nowozin, Instructing people for training gestural interactive systems, in *CHI* (ACM, 2012), pp. 1737–1746
8. HandLogin, http://vip.bu.edu/projects/hcis/hand-login

9. M. Hussein, M. Torki, M.,Gowayyed, M. El-Saban, Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations, in *Proceedings of the 23rd International Joint Conference on Artificial Intelligence* (AAAI Press, 2013), pp. 2466–2472
10. A. Jain, A. Ross, K. Nandakumar, *Introduction to Biometrics* (Springer, Berlin, 2011)
11. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding, in *Proceedings of the ACM International Conference on Multimedia* (ACM, 2014), pp. 675–678
12. S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, H. Winnemoeller, Recognizing image style, in *Proceedings of the British Machine Vision Conference* (BMVA Press, 2014)
13. Kinect for Windows, http://www.microsoft.com/en-us/kinectforwindows
14. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems* (2012), pp. 1097–1105
15. A. Kurakin, Z. Zhang, Z. Liu, A real time system for dynamic hand gesture recognition with a depth sensor, in *Signal Processing Conference (EUSIPCO), 2012 Proceedings of the 20th European* (IEEE, 2012), pp. 1975–1979
16. I. Kviatkovsky, I. Shimshoni, E. Rivlin, Person identification from action styles, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2015), pp. 84–92
17. K. Lai, J. Konrad, P. Ishwar, Towards gesture-based user authentication, in *2012 IEEE Ninth International Conference on Advanced Video and Signal-Based Surveillance* (AVSS, 2012), pp. 282–287. doi:10.1109/AVSS.2012.77
18. W. Li, Z. Zhang, Z. Liu, Action recognition based on a bag of 3D points, in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW, 2010), pp. 9–14. doi:10.1109/CVPRW.2010.5543273
19. T.Y. Lin, A. RoyChowdhury, S. Maji, Bilinear CNN models for fine-grained visual recognition, in *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 1449–1457
20. C. Liu, Beyond pixels: exploring new representations and applications for motion analysis. Ph.D. thesis, Citeseer (2009)
21. L. Miranda, T. Vieira, D. Martinez, T. Lewiner, A. Vieira, M. Campos, Real-time gesture recognition from depth data through key poses learning and decision forests, in *25th SIBGRAPI Conference on Graphics, Patterns and Images* (IEEE, 2012), pp. 268–275
22. M. Müller, T. Röder, M. Clausen, B. Eberhardt, B. Krüger, A. Weber, Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn (2007)
23. F. Ofli, R. Chaudhry, G. Kurillo, R. Vidal, R. Bajcsy, Berkeley mhad: a comprehensive multimodal human action database. IEEE Workshop Appl. Comput. Vis. **0**, 53–60 (2013). doi:10.1109/WACV.2013.6474999
24. O. Oreifej, Z. Liu, Hon4D: histogram of oriented 4D normals for activity recognition from depth sequences, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2013), pp. 716–723
25. E. Park, X. Han, T.L. Berg, A.C. Berg, Combining multiple sources of knowledge in deep CNNs for action recognition, in *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)* (IEEE, 2016), pp. 1–8
26. Z. Ren, J. Yuan, Z. Zhang, Robust hand gesture recognition based on finger-earth mover's distance with a commodity depth camera, in *Proceedings of the 19th ACM International Conference on Multimedia* (ACM, 2011), pp. 1093–1096
27. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein et al., Imagenet large scale visual recognition challenge. Int. J. Comput. Vis. **115**(3), 211–252 (2015)
28. M.J. Scott, M. Niranjan, R.W. Prager, Realisable classifiers: improving operating performance on variable cost problems, in *BMVC* (Citeseer, 1998), pp. 1–10

29. L. Sigal, A.O. Balan, M.J. Black, Humaneva: synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. Int. J. Comput. Vision **87**(1–2), 4–27 (2010). doi:10.1007/s11263-009-0273-6
30. K. Simonyan, A. Zisserman, Two-stream convolutional networks for action recognition in videos, in *Advances in Neural Information Processing Systems* (2014), pp. 568–576
31. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014), arXiv:1409.1556
32. J. Suarez, R. Murphy, Hand gesture recognition with depth images: a review, in *RO-MAN, 2012* (IEEE, 2012), pp. 411–417
33. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9
34. L. Van der Maaten, G. Hinton, Visualizing data using t-SNE. J. Mach. Learn. Res. **9**(2579–2605), 85 (2008)
35. J. Wang, J., Liu, Z., Wu, Y., Yuan, J.: Mining actionlet ensemble for action recognition with depth cameras, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2012), pp. 1290–1297. doi:10.1109/CVPR.2012.6247813
36. J. Wang, Z. Liu, J. Chorowski, Z. Chen, Y. Wu, Robust 3D action recognition with random occupancy patterns, in *Computer Vision–ECCV 2012* (Springer, 2012), pp. 872–885
37. L. Wang, Y. Xiong, Z. Wang, Y. Qiao, Towards good practices for very deep two-stream convnets (2015), arXiv:1507.02159
38. J. Wu, J. Konrad, P. Ishwar, Dynamic time warping for gesture-based user identification and authentication with kinect, in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (ICASSP, 2013), pp. 2371–2375. doi:10.1109/ICASSP.2013.6638079
39. J. Wu, P. Ishwar, J. Konrad, Silhouettes versus skeletons in gesture-based authentication with kinect, in *Proceedings of the IEEE Conference on Advanced Video and Signal-Based Surveillance (AVSS)* (2014)
40. J. Wu, P. Ishwar, J. Konrad, The value of posture, build and dynamics in gesture-based user authentication, in *2014 IEEE International Joint Conference on Biometrics (IJCB)* (IEEE, 2014), pp. 1–8
41. J. Wu, J. Konrad, P. Ishwar, The value of multiple viewpoints in gesture-based user authentication, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW, 2014), pp. 90–97
42. J. Wu, J. Christianson, J. Konrad, P. Ishwar, Leveraging shape and depth in user authentication from in-air hand gestures, in *2015 IEEE International Conference on Image Processing (ICIP)* (IEEE, 2015), pp. 3195–3199
43. L. Xia, C. Chen, J. Aggarwal, View invariant human action recognition using histograms of 3D joints, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW, 2012), pp. 20–27

Chapter 8
DeepGender2: A Generative Approach Toward Occlusion and Low-Resolution Robust Facial Gender Classification via Progressively Trained Attention Shift Convolutional Neural Networks (PTAS-CNN) and Deep Convolutional Generative Adversarial Networks (DCGAN)

Felix Juefei-Xu, Eshan Verma and Marios Savvides

Abstract In this work, we have undertaken the task of occlusion and low-resolution robust facial gender classification. Inspired by the trainable attention model via deep architecture, and the fact that the periocular region is proven to be the most salient region for gender classification purposes, we are able to design a progressive convolutional neural network training paradigm to enforce the attention shift during the learning process. The hope is to enable the network to attend to particular high-profile regions (e.g., the periocular region) without the need to change the network architecture itself. The network benefits from this attention shift and becomes more robust toward occlusions and low-resolution degradations. With the progressively trained attention shift convolutional neural networks (PTAS-CNN) models, we have achieved better gender classification results on the large-scale PCSO mugshot database with 400 K images under occlusion and low-resolution settings, compared to the one undergone traditional training. In addition, our progressively trained network is sufficiently generalized so that it can be robust to occlusions of arbitrary types and at arbitrary locations, as well as low resolution. One way to further improve the robustness of the proposed gender classification algorithm is to invoke a generative approach for occluded image recovery, such as using the deep convolutional

F. Juefei-Xu (✉) · E. Verma · M. Savvides
Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA
e-mail: felixu@cmu.edu

E. Verma
e-mail: everma@cmu.edu

M. Savvides
e-mail: msavvid@ri.cmu.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_8

generative adversarial networks (DCGAN). The facial occlusions degradation studied in this work is a missing data challenge. For the occlusion problems, the missing data locations are known whether they are randomly scattered, or in a contiguous fashion. We have shown, on the PCSO mugshot database, that a deep generative model can very effectively recover the aforementioned degradation, and the recovered images show significant performance improvement on gender classification tasks.

8.1 Introduction

Facial gender classification has always been one of the most studied soft-biometric topics. Over the past decade, gender classification on constrained faces has almost been perfected. However, challenges still remain on less-constrained faces such as faces with occlusions, of low resolution, and off-angle poses. Traditional methods such as the support vector machines (SVMs) and its kernel extension can work pretty well on this classic two-class problem as listed in Table 8.8. In this work, we approach this problem from a very different angle. We are inspired by the booming deep convolutional neural network (CNN) and the attention model to achieve occlusion and low-resolution robust facial gender classification via progressively training the CNN with attention shift. From an orthogonal direction, when images are under severe degradations such as contiguous occlusions, we utilize a deep generative approach to recover the missing pixels such that the facial gender classification performance is further improved on occluded facial images. On one hand, we aim at building a robust gender classifier that is tolerant to image degradations such as occlusions and low resolution, and on the other hand, we aim at mitigating and eliminating the degradations through a generative approach. Together, we stride toward pushing the boundary of unconstrained facial gender classification.

8.1.1 Motivation

Xu et al. [70] proposed an attention-based model that automatically learns to describe the content of images which has been inspired by recent work in machine translation [3] and object detection [2, 57]. In their work, two attention-based image caption generators were introduced under a common framework: (1) a 'soft' deterministic attention mechanism which can be trained by standard back-propagation method and (2) a 'hard' stochastic attention mechanism which can be trained by maximizing an approximate variational lower bound. The encoder of the models uses a convolutional neural network as a feature extractor, and the decoder is composed of a recurrent neural network (RNN) with long short-term memory (LSTM) architecture where the attention mechanism is learned. The authors can then visualize that the network can automatically fix its gaze on the salient objects (regions) in the image while generating the image caption word by word.

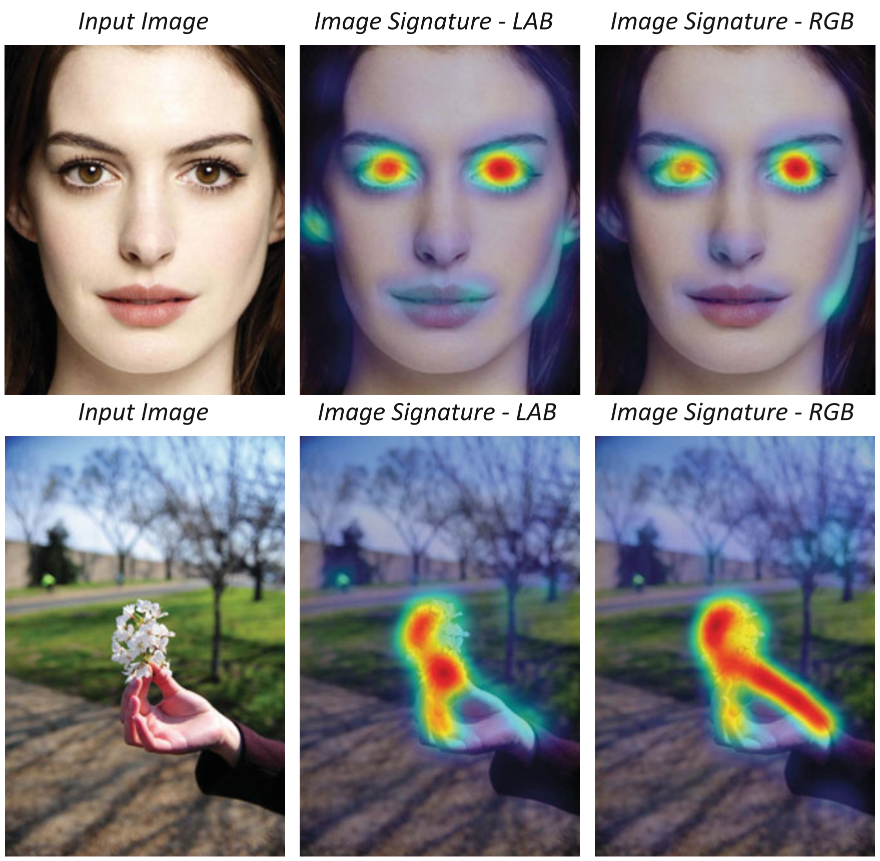


Fig. 8.1 (*Top*) Periocular region on human faces exhibits the highest saliency. (*Bottom*) Foreground object in focus exhibits the highest saliency. Background is blurred with less high-frequency details preserved

For facial gender classification, we know from previous work [16, 56] that the periocular region provides the most important cues for determining the gender information. The periocular region is also the most salient region on human faces, such as shown in the top part of Fig. 8.1, using a general purpose saliency detection algorithm [14]. Similar results can also be obtained using other saliency detection algorithms such as [15, 17]. We can observe from the saliency heat map that the periocular region does fire the most strongly compared to the remainder of the face.

Now we come to think about the following question:

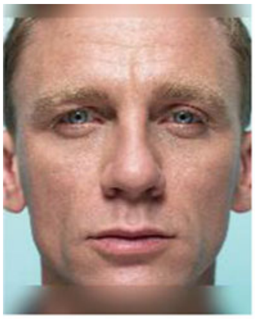
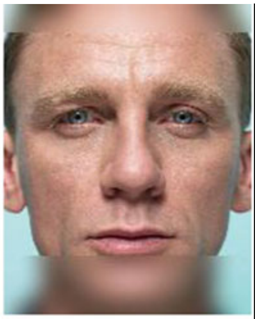
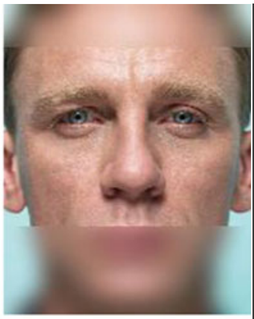
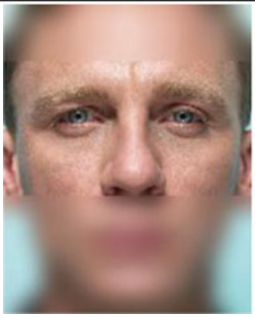
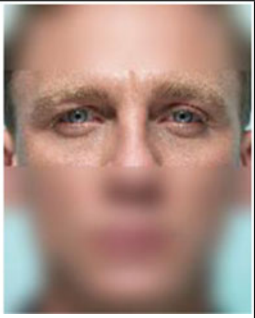
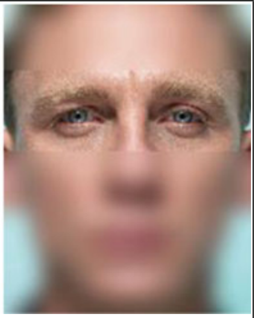
Q: How can we let the CNN shift its attention toward the periocular region, where gender classification has been proven to be the most effective?

The answer comes from our day-to-day experience with photography. If you are using a DSLR camera with a big aperture lens, and fixing the focal point onto an object in the foreground, all background beyond the object in focus will become out of focus and blurred. This is illustrated in the bottom part of Fig. 8.1 and as can be seen, the sharp foreground object (cherry blossom in hand) attracts the most attention in the saliency heat map.

Thus, we can control the attention region by specifying where the image is blurred or remains sharp. In the context of gender classification, we know that we can benefit from fixing the attention onto the periocular region. Therefore, we are 'forcing' what part of the image the network weighs the most, by progressively training the CNN using images with increasing blur levels, zooming into the periocular region, as shown in Table 8.1. Since we still want to use a full-face model, we hope that by employing the mentioned strategy, the learned deep model can be at least on par with other full-face deep models, while harnessing gender cues in the periocular region.

Q: Why not just use the periocular region crop?

Table 8.1 Blurred images with increasing levels of blur

		
13.33%	27.62%	41.90%
		
56.19%	68.57%	73.33%

Although experimentally, periocular is the best facial region for gender classification, we still want to resort to other facial parts (beard/mustache) for providing valuable gender cues. This is especially true when the periocular region is less ideal. For example, some occlusion like sunglasses could be blocking the eye region, and we want our network to still be able to generalize well and perform robustly, even when the periocular region is corrupted.

To strike a good balance between full face-only and periocular-only models, we carry out a progressive training paradigm for CNN that starts with the full face, and progressively zooms into the periocular region by leaving other facial regions blurred. In addition, we hope that the progressively trained network is sufficiently generalized so that it can be robust to occlusions of arbitrary types and at arbitrary locations.

Q: Why blurring instead of blackening out?

We just want to steer the focus, rather than completely eliminating the background, like the DSLR photo example shown in the bottom part of Fig. 8.1. Blackening would create abrupt edges that confuse the filters during the training. When blurred, low-frequency information is still well preserved. One can still recognize the content of the image, e.g., dog, human face, objects, etc. from a blurred image.

Blurring outside the periocular region, and leaving the high-frequency details at the periocular region will both help providing global and structural context of the image, as well as keeping the minute details intact at the region of interest, which will help the gender classification, and fine-grained categorization in general.

Q: Why not let CNN directly learn the blurring step?

We know that CNN filters operate on the entire image, and blurring only part of the image is a pixel location dependent operation and thus is difficult to emulate in the CNN framework. Therefore, we carry out the proposed progressive training paradigm to enforce where the network attention should be.

8.2 Related Work

In this section, we provide relevant background on facial gender classification and attention models.

The periocular region is shown to be the best facial region for recognition purposes [20–22, 24, 27, 31–36, 38, 39, 39, 40, 59, 62, 63], especially for gender classification tasks [16, 56]. A few recent work also applies CNN for gender classification [4, 50]. More related work on gender classification is consolidated in Table 8.8.

Attention models such as the one used for image captioning [70] have gained much popularity only very recently. Rather than compressing an entire image into a static representation, attention allows for salient features to dynamically come to the forefront as needed. This is especially important when there is a lot of clutter in an image. It also helps gaining insight and interpreting the results by visualizing where the model attends to for certain tasks. This mechanism can be viewed as a learnable saliency detector that can be tailored to various tasks, as opposed to the traditional ones such as [8, 14, 15, 17].

It is worth mentioning the key difference between the soft attention and the hard attention. The soft attention is very easy to implement. It produces distribution over input locations, reweights features, and feeds them as input. It can attend to arbitrary input locations using spatial transformer networks [19]. On the other hand, the hard attention can only attend to a single input location, and the optimization cannot utilize gradient descent. The common practice is to use reinforcement learning.

Other applications involving attention models may include machine translation which applies attention over input [54]; speech recognition which applies attention over input sounds [6, 9]; video captioning with attention over input frames [72]; image, question to answer with attention over image itself [69, 75]; and many more [67, 68].

8.3 Proposed Method

Our proposed method involves two major components: a progressively trained attention shift convolutional neural networks (PTAS-CNN) framework for training the unconstrained gender classifier, as well as a deep convolutional generative adversarial networks (DCGAN) for missing pixel recovery on the facial images so that the gender recognition performance can be further improved on images with occlusions.

8.3.1 Progressively Trained Attention Shift Convolutional Neural Networks (PTAS-CNN)

In this section we detail our proposed method on progressively training the CNN with attention shift. The entire training procedure involves $(k + 1)$ epoch groups from epoch group 0 to k, where each epoch group corresponds to one particular blur level.

8.3.1.1 Enforcing Attention in the Training Images

In our experiment, we heuristically choose 7 blur levels, including the one with no blur at all. The example images with increasing blur levels are illustrated in Table 8.1. We use a Gaussian blur kernel with $\sigma = 7$ to blur the corresponding image regions.

Doing this is conceptually enforcing the network attention in the training images without the need of changing the network architecture.

8.3.1.2 Progressive CNN Training with Attention

We employ the AlexNet [46] architecture for our progressive CNN training. The AlexNet has 60 million parameters and 650,000 neuron, consisting of 5 convolution layers and 3 fully connected layers with a final 1000-way softmax. To reduce overfitting in the fully connected layers, AlexNet employs "dropout" and data augmentation, both of which are preserved in our training. The main difference is that we only need a 2-way softmax due to the nature of gender classification problems.

As illustrated in Fig. 8.2, the progressive CNN training begins with the first epoch group (Epoch Group 0, images with no blur), and the first CNN model $\mathcal{M}_0$ is obtained and frozen after convergence. Then, we input the next epoch group for tuning the $\mathcal{M}_0$ and in the end produce the second model $\mathcal{M}_1$, with attention enforced through training images. The procedure is carried out sequentially until the final model $\mathcal{M}_k$ is obtained. Each $\mathcal{M}_j (j = 0, \ldots, k)$ is trained with 1000 epochs and with a batchsize of 128. At the end of the training for step j, the model corresponding to best validation accuracy is taken ahead to the next iteration $(j + 1)$.

8.3.1.3 Implicit Low-Rank Regularization in CNN

Blurring the training images in our paradigm may have more implications. Here we want to show the similarities between low-pass Fourier analysis and low-rank approximation in SVD. Through the analysis, we hope to make connections to the low-rank regularization procedure in the CNN. We have learned from a recent work [65] that enforcing a low-rank regularization and removing the redundancy in the convolution kernels is important and can help improve both the classification accuracy and the computation speed. Fourier analysis involves expansion of the original data x_{ij} (taken from the data matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$) in an orthogonal basis, which is the inverse Fourier transform:

$$x_{ij} = \frac{1}{m}\sum_{k=0}^{m-1} c_k e^{\mathbf{i}2\pi jk/m}. \tag{8.1}$$

The connection with SVD can be explicitly illustrated by normalizing the vector $\{e^{\mathbf{i}2\pi jk/m}\}$ and by naming it $\mathbf{v}'_k$:

$$x_{ij} = \sum_{k=0}^{m-1} b_{ik} v'_{jk} = \sum_{k=0}^{m-1} u'_{ik} s'_k v'_{jk}, \tag{8.2}$$

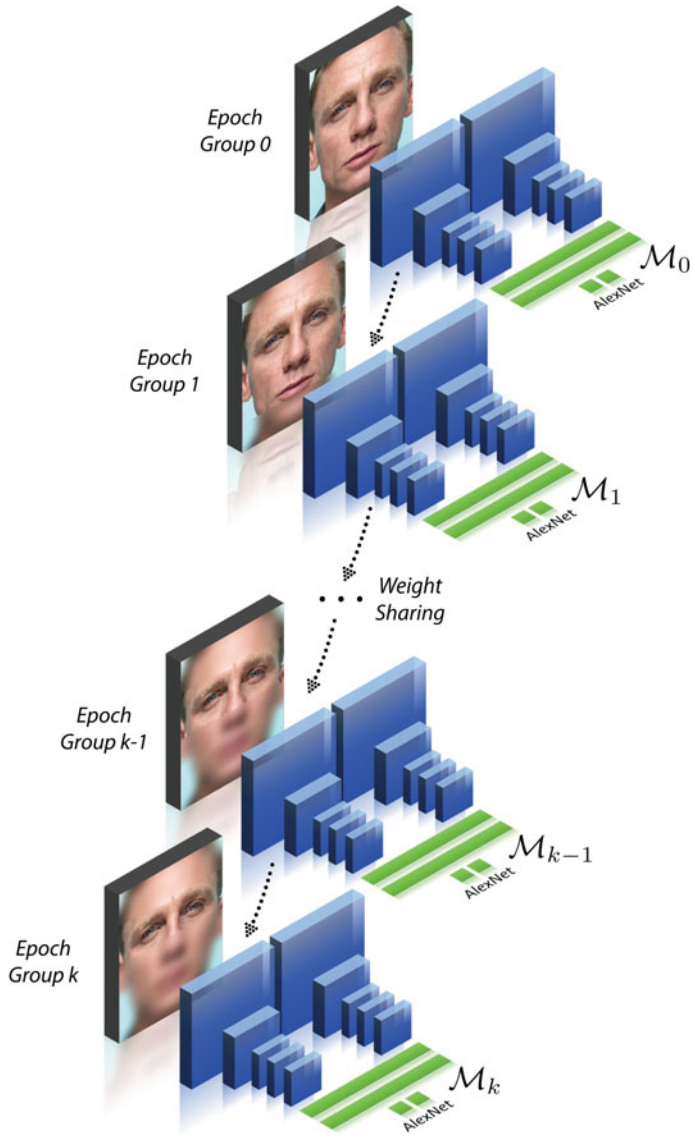


Fig. 8.2 Progressive CNN training with attention

which generates the matrix equation $\mathbf{X} = \mathbf{U}'\boldsymbol{\Sigma}'\mathbf{V}'^{\top}$. However, unlike the SVD, even though the $\{\mathbf{v}'_k\}$ are an orthonormal basis, the $\{\mathbf{u}'_k\}$ are not in general orthogonal. Nevertheless this demonstrates how the SVD is similar to a Fourier transform. Next, we will show that the **low-pass filtering in Fourier analysis** is closely related to the **low-rank approximation in SVD**.

Suppose we have N image data samples of dimension d in the original two-dimensional form $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Let matrix $\hat{\mathbf{X}}$ contain all the data samples undergone 2D Fourier transform $\mathcal{F}(\cdot)$, in the vectorized form:

$$\hat{\mathbf{X}} = \begin{bmatrix} | & | & & | \\ \text{vec}(\mathcal{F}(\mathbf{x}_1)) & \text{vec}(\mathcal{F}(\mathbf{x}_2)) & \ldots & \text{vec}(\mathcal{F}(\mathbf{x}_N)) \\ | & | & & | \end{bmatrix}_{d\times N}.$$

Matrix $\hat{\mathbf{X}}$ can be decomposed using SVD: $\hat{\mathbf{X}} = \hat{\mathbf{U}}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{V}}^{\top}$. Without loss of generality, let us assume that $N = d$ for brevity. Let $\mathbf{g}$ and $\hat{\mathbf{g}}$ be the Gaussian filter in spatial domain and frequency domain, respectively, namely $\hat{\mathbf{g}} = \mathcal{F}(\mathbf{g})$. Let $\hat{\mathbf{G}}$ be a diagonal matrix with $\hat{\mathbf{g}}$ on its diagonal. The convolution operation becomes dot product in frequency domain, so the blurring operation becomes

$$\hat{\mathbf{X}}_{\text{blur}} = \hat{\mathbf{G}} \cdot \hat{\mathbf{X}} = \hat{\mathbf{G}} \cdot \hat{\mathbf{U}}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{V}}^{\top}, \tag{8.3}$$

where $\hat{\boldsymbol{\Sigma}} = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_d)$ contains the singular values of $\hat{\mathbf{X}}_{\text{blur}}$, already sorted in descending order: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_d$. Suppose we can find a permutation matrix $\mathbf{P}$ such that when applied on the diagonal matrix $\hat{\mathbf{G}}$, the diagonal elements are sorted in descending order according to the magnitude: $\hat{\mathbf{G}}' = \mathbf{P}\hat{\mathbf{G}} = \text{diag}(\hat{g}'_1, \hat{g}'_2, \ldots, \hat{g}'_d)$. Now, let us apply the same permutation operation on $\hat{\mathbf{X}}_{\text{blur}}$, we can thus have the following relationship:

$$\mathbf{P} \cdot \hat{\mathbf{X}}_{\text{blur}} = \mathbf{P} \cdot \hat{\mathbf{G}} \cdot \hat{\mathbf{U}}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{V}}^{\top} \tag{8.4}$$

$$\hat{\mathbf{X}}'_{\text{blur}} = \hat{\mathbf{G}}' \cdot \hat{\mathbf{U}}\hat{\boldsymbol{\Sigma}}\hat{\mathbf{V}}^{\top} = \hat{\mathbf{U}} \cdot (\hat{\mathbf{G}}'\hat{\boldsymbol{\Sigma}}) \cdot \hat{\mathbf{V}}^{\top} \tag{8.5}$$

$$= \hat{\mathbf{U}} \cdot \text{diag}(\hat{g}'_1\sigma_1, \hat{g}'_2\sigma_2, \ldots, \hat{g}'_d\sigma_d) \cdot \hat{\mathbf{V}}^{\top}. \tag{8.6}$$

Due to the fact that Gaussian distribution is not a heavy-tailed distribution, the already smaller singular values will be brought down to 0 by the Gaussian weights. Therefore, $\hat{\mathbf{X}}_{\text{blur}}$ actually becomes low-rank after Gaussian low-pass filtering. To this end, we can say that low-pass filtering in Fourier analysis is equivalent to the low-rank approximation in SVD up to a permutation.

This phenomenon is loosely observed through the visualization of the trained filters, as shown in Fig. 8.15, which will be further analyzed and studied in future work.

8.3.2 Occlusion Removal via Deep Convolutional Generative Adversarial Networks (DCGAN)

In this section, we first review the basics of DCGAN and then show how DCGAN can be utilized for occlusion removal, or missing pixel recovery on face images.

8.3.2.1 Deep Convolutional Generative Adversarial Networks

The generative adversarial network (GAN) [12] is capable of generating high-quality images. The framework trains two networks, a generator $\mathcal{G}_\theta(\mathbf{z}) : \mathbf{z} \rightarrow \mathbf{x}$, and a discriminator $\mathcal{D}_\omega(\mathbf{x}) : \mathbf{x} \rightarrow [0, 1]$. $\mathcal{G}$ maps a random vector $\mathbf{z}$, sampled from a prior distribution $p_\mathbf{z}(\mathbf{z})$, to the image space. $\mathcal{D}$ maps an input image to a likelihood. The purpose of $\mathcal{G}$ is to generate realistic images, while $\mathcal{D}$ plays an adversarial role to discriminate between the image generated from $\mathcal{G}$, and the image sampled from data distribution p_{data}. The networks are trained by optimizing the following minimax loss function:

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{G}, \mathcal{D}) = \mathbb{E}_{\mathbf{x}\sim p_{\text{data}}(\mathbf{x})}\Big[\log(\mathcal{D}(\mathbf{x}))\Big] + \mathbb{E}_{\mathbf{z}\sim p_\mathbf{z}(\mathbf{z})}\Big[\log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z}))\Big],$$

where $\mathbf{x}$ is the sample from the p_{data} distribution; $\mathbf{z}$ is randomly generated and lies in some latent space. There are many ways to structure $\mathcal{G}(\mathbf{z})$. The deep convolutional generative adversarial network (DCGAN) [60] uses fractionally strided convolutions to upsample images instead of fully connected neurons as shown in Fig. 8.3.

The generator $\mathcal{G}$ is updated to fool the discriminator $\mathcal{D}$ into wrongly classifying the generated sample, $\mathcal{G}(\mathbf{z})$, while the discriminator $\mathcal{D}$ tries not to be fooled. In this work, both $\mathcal{G}$ and $\mathcal{D}$ are deep convolutional neural networks and are trained with an alternating gradient descent algorithm. After convergence, $\mathcal{D}$ is able to reject images that are too fake, and $\mathcal{G}$ can produce high-quality images faithful to the training distribution (true distribution p_{data}).

8.3.2.2 Occlusion Removal via DCGAN

To take on the missing data challenge, we need to utilize both the $\mathcal{G}$ and $\mathcal{D}$ networks from DCGAN, pre-trained with uncorrupted data. After training, $\mathcal{G}$ is able to embed the images from p_{data} onto some nonlinear manifold of $\mathbf{z}$. An image that is not from p_{data} (e.g., corrupted face image with missing pixels) should not lie on the learned manifold. Therefore, we seek to recover the "closest" image on the manifold to the corrupted image as the proper reconstruction.

Let us denote the corrupted image as $\mathbf{y}$. To quantify the "closest" mapping from $\mathbf{y}$ to the reconstruction, we define a function consisting of contextual loss and perceptual loss, following the work of Yeh et al. [73].

In order to incorporate the information from the uncorrupted portion of the given image, the **contextual loss** is used to measure the fidelity between the reconstructed image portion and the uncorrupted image portion, which is defined as

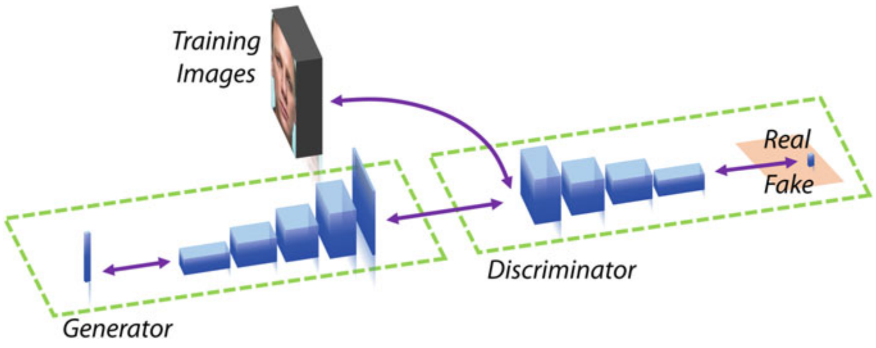


Fig. 8.3 Pipeline of a standard DCGAN with the generator $\mathcal{G}$ mapping a random vector $\mathbf{z}$ to an image and the discriminator $\mathcal{D}$ mapping the image (from true distribution or generated) to a probability value

$$\mathcal{L}_{\text{contextual}}(\mathbf{z}) = \|\mathbf{M} \odot \mathcal{G}(\mathbf{z}) - \mathbf{M} \odot \mathbf{y}\|_1, \tag{8.7}$$

where $\mathbf{M}$ denotes the binary mask of the uncorrupted region and $\odot$ denotes the element-wise Hadamard product operation. The corrupted portion, i.e., $(1 - \mathbf{M}) \odot \mathbf{y}$, is not used in the loss. The choice of ℓ_1-norm is empirical. From the experiments carried out in [73], images recovered with ℓ_1-norm loss tend to be sharper and with higher quality compared to ones reconstructed with ℓ_2-norm.

The **perceptual loss** encourages the reconstructed image to be similar to the samples drawn from the training set (true distribution p_{data}). This is achieved by updating $\mathbf{z}$ to fool $\mathcal{D}$, or equivalently to have a high value of $\mathcal{D}(\mathcal{G}(\mathbf{z}))$. As a result, $\mathcal{D}$ will predict $\mathcal{G}(\mathbf{z})$ to be from the data with a high probability. The same loss for fooling $\mathcal{D}$ as in DCGAN is used:

$$\mathcal{L}_{\text{perceptual}}(\mathbf{z}) = \log(1 - \mathcal{D}(\mathcal{G}(\mathbf{z}))). \tag{8.8}$$

The corrupted image with missing pixels can now be mapped to the closest $\mathbf{z}$ in the latent representation space with the defined perceptual and contextual losses. We follow the training procedure in [60] and use Adam [45] for optimization. $\mathbf{z}$ is updated using back-propagation with the total loss:

$$\hat{\mathbf{z}} = \arg\min_{\mathbf{z}}(\mathcal{L}_{\text{contextual}}(\mathbf{z}) + \lambda \mathcal{L}_{\text{perceptual}}(\mathbf{z})) \tag{8.9}$$

where λ is a weighting parameter. After finding the optimal solution $\hat{\mathbf{z}}$, the hallucinated full-face image can be obtained by

$$\mathbf{x}_{\text{hallucinated}} = \mathbf{M} \odot \mathbf{y} + (1 - \mathbf{M}) \odot \mathcal{G}(\hat{\mathbf{z}}). \tag{8.10}$$

Examples of face recovery from contiguous occlusions are shown in Fig. 8.11 using DCGAN. Applying this deep generative approach for occluded image recovery is expected to improve the performance of unconstrained gender classification.

8.4 Experiments: Part A

In this section we detail the training and testing protocols employed and various occlusions and low resolutions modeled in the testing set. Accompanying figures and tables for each subsection encompass the results and observations and are elaborated in each section.[1]

8.4.1 Database and Preprocessing

Training set: We source images from 5 different datasets, each containing samples of both classes. The datasets are J-Mugshot, O-Mugshot, M-Mugshot, P-Mugshot, and Pinellas. All the datasets, except Pinellas, are evenly separated into males and females of different ethnicities. The images are centered, by which, we mean that we have landmarked certain points on the face, which are then anchored to fixed points in the resulting training image. For example, the eyes are anchored at the same coordinates in every image. All of our input images have the same dimension 168×210. The details of the training datasets are listed in Table 8.2. The images are partitioned into training and validation and the progressive blur is applied to each image as explained in the previous section. Hence, for a given model iteration, the training set consists of $\sim$90 k images.

Testing set: The testing set was built primarily from the following two datasets: (1) The AR Face database [55] is one of the most widely used face databases with occlusions. It contains 3,288 color images from 135 subjects (76 male subjects + 59 female subjects). Typical occlusions include sunglasses and scarves. The database also captures expression variations and lighting changes. (2) Pinellas County Sherrif's Office (PCSO) mugshot database is a large-scale database of over 1.4 million images. We took a subset of around 400 K images from this dataset. These images are not seen during training.

The testing images are centered and cropped in the same way as the training images, though other preprocessing like the progressive blur are not applied. Instead, to model real world occlusions we have conducted the following experiments to be discussed in Sect. 8.4.2.

[1]**A note on legend**: (1) Symbols $\mathcal{M}$ correspond to each model trained, with $\mathcal{M}_F$ corresponding to the model trained on full face (equivalent to $\mathcal{M}_0$), $\mathcal{M}_P$ to one with just periocular images and $\mathcal{M}_k$, $k \subseteq (1, \ldots, 6)$ to the incremental models trained. (2) The tabular results show model performance on the original images in column 1 and corrupted images in other columns.

Table 8.2 Datasets used for progressive CNN training

Database	Males	Females
J-Mugshot	1900	1371
M-Mugshot	1772	805
Pinellas subset	13215	3394
P-Mugshot	46346	12402
O-Mugshot	4164	3634
	67397	21606
Total		89003

8.4.2 *Experiment I: Occlusion Robustness*

In Experiment I, we carry out occlusion robust gender classification on both the AR Face database and the PCSO mugshot database. We manually add artificial occlusions to test the efficacy of our method on the PCSO database and test on the various images sets in the AR Face dataset.

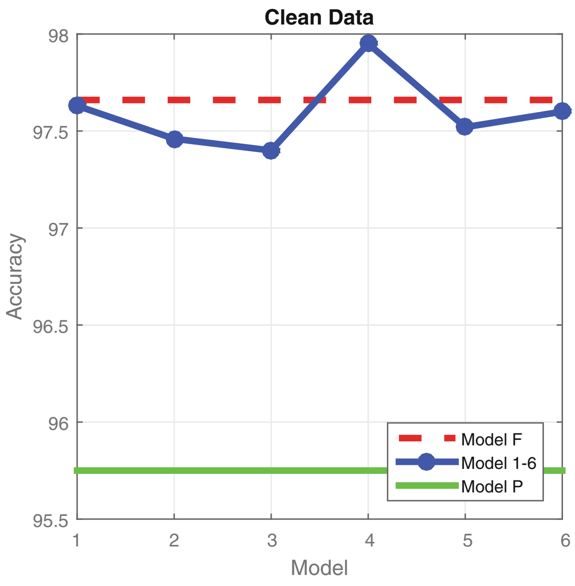


Fig. 8.4 Overall classification accuracy on the PCSO (400 K). Images are not corrupted

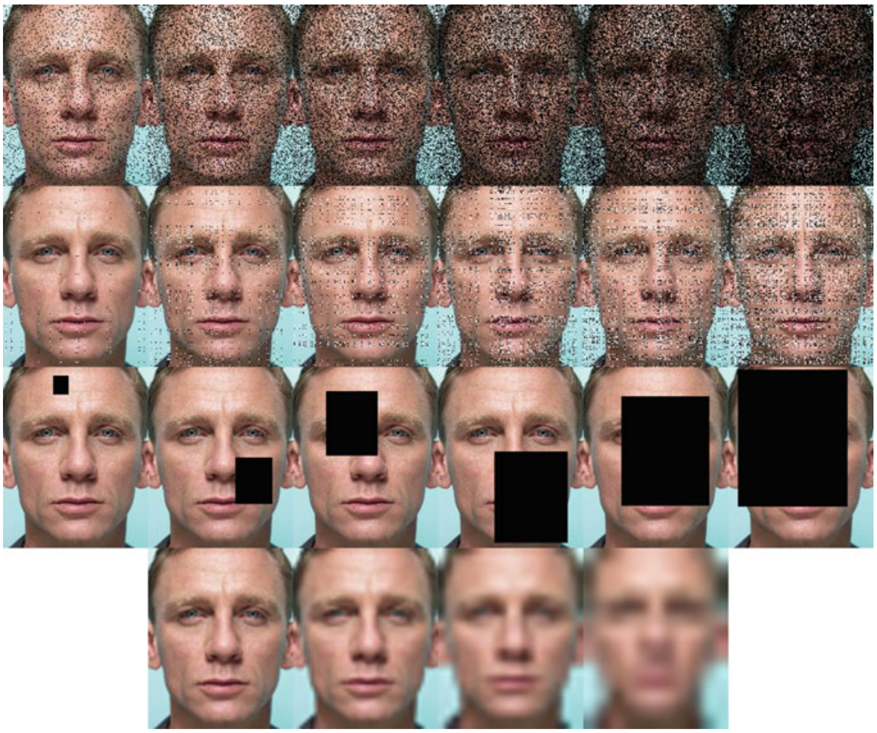


Fig. 8.5 Various degradations applied on the testing images for Experiment I and II. Row 1 random missing pixel occlusions; Row 2 random additive Gaussian noise occlusions; Row 3 random contiguous occlusions. Percentage of degradation for Row 1–3 10, 25, 35, 50, 65 and 75%. Row 4 various zooming factors (2x, 4x, 8x, 16x) for low-resolution degradations

8.4.2.1 Experiments on the PCSO Mugshot Database

To begin with, the performance of various models on the clean PCSO data is shown in Fig. 8.4. As expected, if the testing images are clean, it should be preferable to use $\mathcal{M}_F$, rather than $\mathcal{M}_P$. We can see that the progressively trained models $\mathcal{M}_1 - \mathcal{M}_6$ are on par with $\mathcal{M}_F$.

We corrupt the testing images (400 K) with three types of facial occlusions. These are visualized in Fig. 8.5 with each row corresponding to some modeled occlusions.

(1) Random Missing Pixels Occlusions

Table 8.3 Overall classification accuracy on the PCSO (400 K). Images are corrupted with **random missing pixels** of various percentages

Corruption	0%	10%	25%	35%	50%	65%	75%
$\mathcal{M}_F$	97.66	97.06	93.61	89.15	82.39	79.46	77.40
$\mathcal{M}_1$	97.63	96.93	92.68	87.99	81.57	78.97	77.20
$\mathcal{M}_2$	97.46	96.83	93.19	89.17	83.03	80.06	77.68
$\mathcal{M}_3$	97.40	96.98	94.06	90.65	84.79	81.59	78.56
$\mathcal{M}_4$	97.95	97.63	95.63	93.10	87.96	84.41	80.22
$\mathcal{M}_5$	97.52	97.26	95.80	94.07	90.40	87.39	83.04
$\mathcal{M}_6$	97.60	97.29	95.50	93.27	88.80	85.57	81.42
$\mathcal{M}_P$	95.75	95.45	93.84	92.02	88.87	86.59	83.18

Varying factors of the image pixels (10, 25, 35, 50, 65 and 75%) were dropped to model lost information and grainy images.[2] This is corresponding to the first row in Fig. 8.5. From Table 8.3 and Fig. 8.6, $\mathcal{M}_5$ performs the best with $\mathcal{M}_6$ showing a dip in accuracy suggesting a tighter periocular region is not well suited for such applications, i.e., a limit on the periocular region needs to be maintained in the blur set. There is a flip in performance of the models $\mathcal{M}_P$ and $\mathcal{M}_F$ going from the original to 25% with the periocular model generalizing better for higher corruptions. As the percentage of missing pixels increases, the performance gap between $\mathcal{M}_P$ and $\mathcal{M}_F$ increases. As hypothesized, the trend of improving performance between progressively trained models is maintained across factors indicating a better learned model toward noise.

(2) Random Additive Gaussian Noise Occlusions

Gaussian white noise ($\sigma = 6$) was added to image pixels in varying factors (10, 25, 35, 50, 65 and 75%). This is corresponding to the second row in Fig. 8.5 and is done to model high noise data and bad compression. From Table 8.4 and Fig. 8.7, $\mathcal{M}_4 - \mathcal{M}_6$ perform best for medium noise. For high noise, $\mathcal{M}_5$ is the most robust. Just like before, as the noise increases, the trend undertaken by the performance of $\mathcal{M}_P$ & $\mathcal{M}_F$ and $\mathcal{M}_5$ & $\mathcal{M}_6$ is maintained and so is the performance trend of the progressively trained models.

(3) Random Contiguous Occlusions

[2]This can also model the dead pixel/shot noise of a sensor and these results can be used to accelerate inline gender detection by using partially demosaiced images.

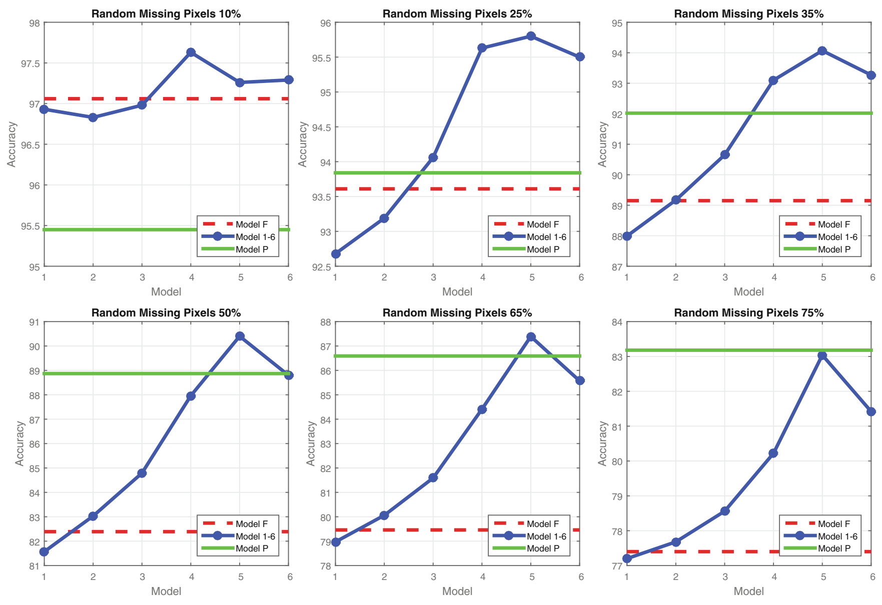


Fig. 8.6 Overall classification accuracy on the PCSO (400 K). Images are corrupted with **random missing pixels** of various percentages

Table 8.4 Overall classification accuracy on the PCSO (400 K). Images are corrupted with **additive Gaussian random noise** of various percentages

Corruption	0%	10%	25%	35%	50%	65%	75%
$\mathcal{M}_F$	97.66	97.00	94.03	91.19	86.47	83.43	79.94
$\mathcal{M}_1$	97.63	96.93	94.00	91.26	87.00	84.27	81.15
$\mathcal{M}_2$	97.46	96.87	94.43	92.19	88.75	86.44	83.33
$\mathcal{M}_3$	97.40	97.00	95.18	93.27	89.93	87.55	84.16
$\mathcal{M}_4$	97.95	97.67	96.45	95.11	92.43	90.28	87.06
$\mathcal{M}_5$	97.52	97.29	96.25	95.21	93.21	91.65	89.12
$\mathcal{M}_6$	97.60	97.32	96.04	94.77	92.46	90.80	88.08
$\mathcal{M}_P$	95.75	95.59	94.85	94.00	92.43	91.15	88.74

To model big occlusions like sunglasses or other contiguous elements, continuous patches of pixels (10, 25, 35, 50, 65 and 75%) were dropped from the image as seen in the third row of Fig. 8.5. The most realistic occlusion corresponds to the first few patches, and other patches are extreme cases. For the former cases, $\mathcal{M}_1 - \mathcal{M}_3$ are able to predict the classes with the highest accuracy. From Table 8.5 and Fig. 8.8, for such large occlusions and missing data, more contextual information is needed for correct classification since $\mathcal{M}_1 - \mathcal{M}_3$ perform better than other models.

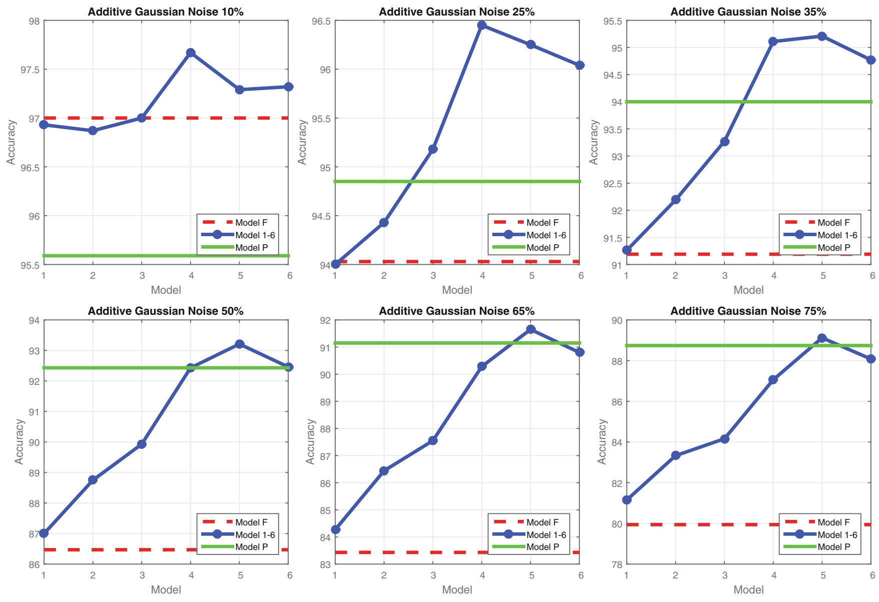


Fig. 8.7 Overall classification accuracy on the PCSO (400 K). Images are corrupted with **additive Gaussian random noise** of various percentages

Table 8.5 Overall classification accuracy on the PCSO (400 K). Images are corrupted with **random contiguous occlusions** of various percentages

Corruption	0%	10%	25%	35%	50%	65%	75%
$\mathcal{M}_F$	97.66	96.69	93.93	88.63	76.54	73.75	64.82
$\mathcal{M}_1$	97.63	96.95	94.64	90.20	77.47	75.20	53.04
$\mathcal{M}_2$	97.46	96.76	94.56	90.04	75.99	70.83	56.25
$\mathcal{M}_3$	97.40	96.63	94.65	90.08	77.13	71.77	68.52
$\mathcal{M}_4$	97.95	96.82	92.70	86.64	75.25	70.37	61.63
$\mathcal{M}_5$	97.52	96.56	92.03	83.95	70.36	69.94	66.52
$\mathcal{M}_6$	97.60	96.61	93.08	86.34	71.91	71.40	69.50
$\mathcal{M}_P$	95.75	95.00	93.01	88.34	76.82	67.81	49.73

However, since they perform better than $\mathcal{M}_F$, our scheme of focused saliency helps generalizing over occlusions.

8.4.2.2 Experiments on the AR Face Database

We partitioned the original set to smaller subsets to better understand our methodology's performance under different conditions. Set 1 consists of neutral expression,

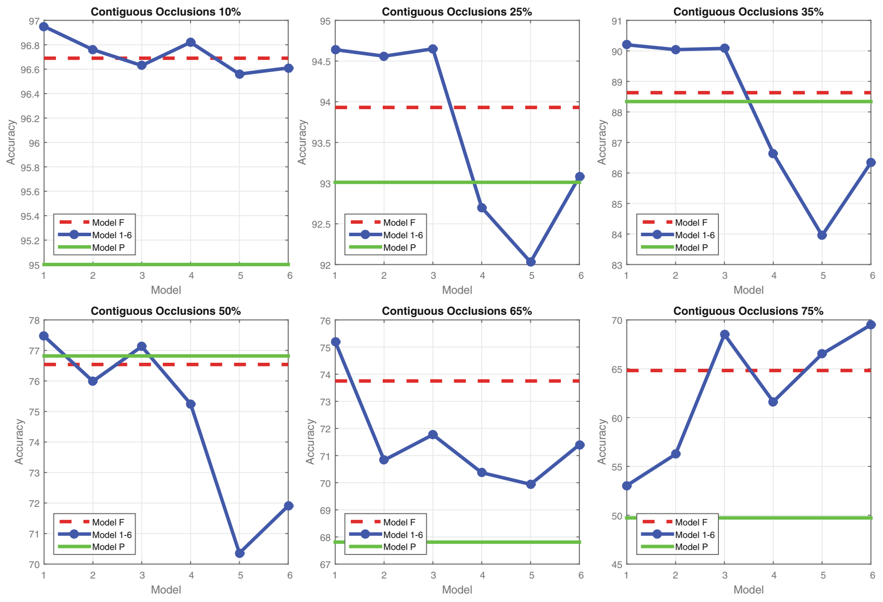


Fig. 8.8 Overall classification accuracy on the PCSO (400 K). Images are corrupted with **random contiguous occlusions** of various percentages

full-face subjects. Set 2 has full face but varied expressions. Set 3 includes periocular occlusions such as sunglasses and Set 4 includes these and other occlusions like clothing, etc. Set 5 is the entire dataset including illumination variations.

Referring to Table 8.6 and Fig. 8.9, for Set 1, the full-face model performs the best and this is expected as this model was trained on images very similar to this. Set 2 suggests that the models need more contextual information when expressions are introduced. Thus, $\mathcal{M}_4$ which has focus on periocular but has face information too performs best. For Set 3, we can see two things: one, $\mathcal{M}_P$ performs better than $\mathcal{M}_F$ indicative of its robustness to periocular occlusions. Two, $\mathcal{M}_5$ is the best as it combines periocular focus with contextual information gained from incremental training.

Set 4 performance brings out why periocular region is preferred for occluded faces. We ascertained that some texture and loss of face contour is throwing off the models $\mathcal{M}_1 - \mathcal{M}_6$. The performance of the models on Set 5 reiterates previously stated observations of the combined importance of contextual information about face contours and the importance of periocular region. This is the reason for the best accuracy reported by $\mathcal{M}_3$.

Table 8.6 Gender classification accuracy on the AR Face database

Sets	Set 1	Set 2	Set 3	Set 4	Set 5 (Full Set)
$\mathcal{M}_F$	98.44	93.23	89.06	83.04	81.65
$\mathcal{M}_1$	97.66	92.71	86.72	81.70	82.82
$\mathcal{M}_2$	97.66	92.71	90.62	82.14	85.10
$\mathcal{M}_3$	97.66	93.23	91.41	80.80	85.62
$\mathcal{M}_4$	98.44	95.31	92.97	77.23	84.61
$\mathcal{M}_5$	96.88	93.49	94.53	80.36	84.67
$\mathcal{M}_6$	96.09	92.71	92.97	79.02	83.90
$\mathcal{M}_P$	96.09	90.62	91.41	86.61	83.44

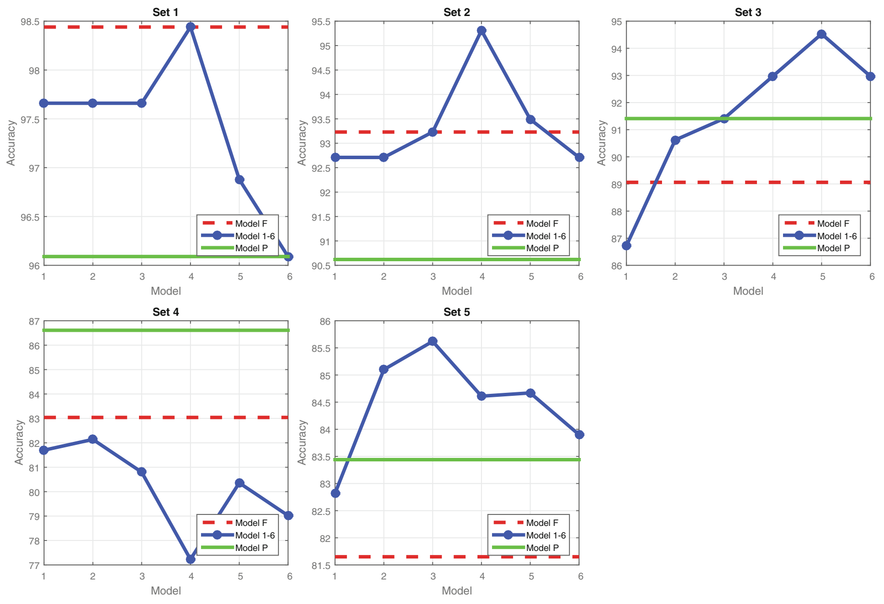


Fig. 8.9 Gender classification accuracy on the AR Face database

8.4.3 Experiment II: Low-Resolution Robustness

Our scheme of training on Gaussian blurred images should generalize well to low-resolution images. To test this hypothesis, we tested our models on images from the PCSO mugshot dataset by first down-sampling them by a factor and then blowing them back up (zooming factor for example: 2x, 4x, 8x, 16x).[3] This inculcates the loss of edge information and other higher order information and is captured in the

[3]Effective pixel for 16x zooming factor is around 10×13, which is a quite challenging low-resolution setting.

Table 8.7 Overall classification accuracy on the PCSO (400 K). Images are down-sampled to a **lower resolution** with various zooming factors

Zooming factor	1x	2x	4x	8x	16x
$\mathcal{M}_F$	97.66	97.55	96.99	94.19	87.45
$\mathcal{M}_1$	97.63	97.48	96.91	94.76	87.41
$\mathcal{M}_2$	97.46	97.31	96.73	94.77	88.82
$\mathcal{M}_3$	97.40	97.20	96.37	93.50	87.57
$\mathcal{M}_4$	97.95	97.89	97.56	95.67	90.17
$\mathcal{M}_5$	97.52	97.40	96.79	95.26	89.66
$\mathcal{M}_6$	97.60	97.51	97.05	95.42	90.79
$\mathcal{M}_P$	95.75	95.65	95.27	94.12	91.59

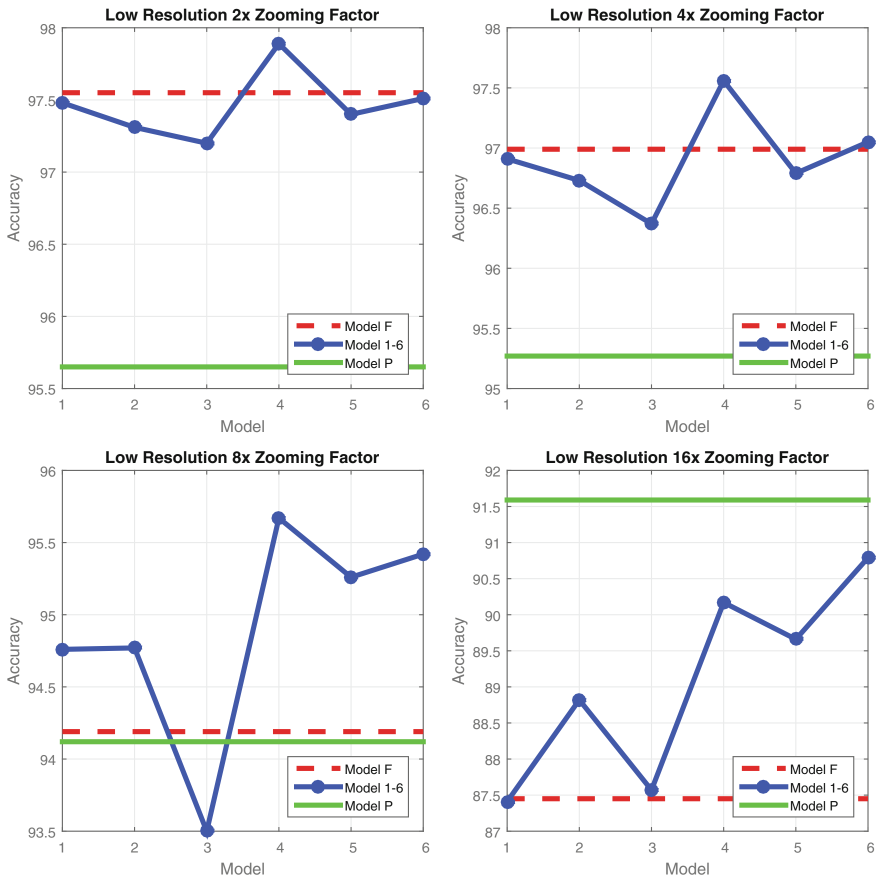


Fig. 8.10 Overall classification accuracy on the PCSO (400 K). Images are down-sampled to a **lower resolution** with various zooming factors

last row of Fig. 8.5. As seen in Table 8.7 and Fig. 8.10 for cases, 2x, 4x, 8x, the trend between $\mathcal{M}_1 - \mathcal{M}_6$ and their performance with respect to $\mathcal{M}_F$ is maintained. As mentioned before, $\mathcal{M}_4$ performs well due to the balance between focus on periocular region and saving the contextual information of a face.

8.4.4 Discussion

We have proposed a methodology for building a gender recognition system which is robust to occlusions. It involves training a deep model incrementally over several batches of input data preprocessed with progressive blur. The intuition and intent is twofold, one to have the network focus on periocular regions of the face for gender recognition. And two, to preserve contextual information of facial contours to generalize better over occlusions.

Through various experiments we have observed that our hypothesis is indeed true and that for a given occlusion set, it is possible to have high accuracy from a model that encompasses both of above-stated properties. Irrespective of the fact that we did not train on any occluded data, or optimize for a particular set of occlusions, our models are able to generalize well over synthetic data and real-life facial occlusion images.

We have summarized the overall experiments and consolidated the results in Table 8.8. For PCSO large-scale experiments, we believe that 35% occlusion is the right amount of degradations, on which accuracies should be reported. Therefore, we average the accuracy from our best model on three types of occlusions (missing pixel, additive Gaussian noise, and contiguous occlusions) which gives 93.12% in Table 8.8. For low-resolution experiments, we believe 8x zooming factor is the right amount of degradations, so we report the accuracy 95.67% in Table 8.8. Many other related work on gender classification are also listed for a quick comparison. This table is based on [16].

8.5 Experiments: Part B

In order to boost the classification accuracy and support the models trained in the previous section, we trained a GAN to hallucinate occlusions and missing information in the input image. The next two sections detail our curation of the input data and the selection of the testing sets for our evaluation. The gender model ($\mathcal{M}_k$) definitions remain the same as in the previous section. And we use $\mathcal{G}_z$ and $\mathcal{D}_x$ to denote the Generator and Discriminator models from GAN.

Table 8.8 Summary of many related work on gender classification. The proposed method is shown in the top rows

Authors	Methods	Dataset	Variation	Unique Subj.	Resolution	# of Subjects (Male/Female)	Accuracy
Proposed	Progressive CNN training w/ attention	Mugshots	Frontal-only, mugshot	Yes	168×210	89k total tr, 400k total te	97.95% te
			Occlusion	Yes		89k total tr, 400k total te	93.12% te
			Low-resolution	Yes		89k total tr, 400k total te	95.67% te
		AR Face	Expr x4, occl x2	No		89k total tr, 76/59 te	85.62% te
Hu et al. [16]	Region-based MR-8 filter bank w/ fusion of linear SVMs	Flickr	Li,exp,pos,bkgd,occl	Yes	128×170	10037/10037 tr, 3346/3346 te	90.1% te
		FERET	Frontal-only, studio	Yes		320/320 tr, 80/80 te	92.8% te
Chen and Lin [7]	Color and edge features w/ Adaboost+weak classifiers	Web img	Lighting, expression background	Yes	N/A	1948 total tr, 210/259 te	87.6% te

(continued)

Table 8.8 (continued)

Authors	Methods	Dataset	Variation	Unique Subj.	Resolution	# of Subjects (Male/Female)	Accuracy
Wang et al. [10]	Gabor filters w/ polynomial-SVM	BioID	Lighting and expression	Yes	286 × 384	976/544 tr, 120 total te	92.5% te
Golomb et al. [11]	Raw pixel w/ neural network	SexNet	Frontal-only, studio	Yes	30 × 30	40/40 tr, 5/5 te	91.9% tr
Gutta et al. [13]	Raw pixel w/ mix of neural net RBF-SVM & decision tree	FERET	Frontal-only, studio	No	64 × 72	1906/1100 tr, 47/30 te	96.0% te
Jabid et al. [18]	Local directional patterns w/ SVM	FERET	Frontal-only, studio	No	100 × 100	1100/900 tr, unknown te	95.1% te
Lee et al. [48]	Region-based w/ linear regression fused w/ SVM	FERET	Frontal-only, studio	N/A	N/A	1158/615 tr, unknown te	98.8% te
		Web img	Unknown			1500/1500 tr, 1500/1500 te	88.1% te

(continued)

Table 8.8 (continued)

Authors	Methods	Dataset	Variation	Unique Subj.	Resolution	# of Subjects (Male/Female)	Accuracy
Leng and Wang [49]	Gabor filters w/ fuzzy-SVM	FERET	Frontal-only, studio	Yes	256×384	160/140 total, 80% tr, 20% te	98.1% te
		CAS-PEAL	Studio	N/A	140×120	400/400 total, 80% tr, 20% te	93.0% te
		BUAA-IRIP	Frontal-only, studio	No	56×46	150/150 total, 80% tr, 20% te	89.0% te
Lin et al. [51]	Gabor filters w/ linear SVM	FERET	Frontal-only, studio	N/A	48×48	Unknown	92.8% te
Lu & Lin [52]	Gabor filters w/ Adaboost + linear SVM	FERET	Frontal-only, studio	N/A	48×48	150/150 tr, 518 total te	90.0% te
Lu et al. [53]	Region-based w/ RBF-SVM fused w/ majority vote	CAS-PEAL	Studio	Yes	90×72	320/320 tr, 80/80 te	92.6% te
Moghaddam and Yang [58]	Raw pixel w/ RBF-SVM	FERET	Frontal-only, studio	N/A	80×40	793/715 tr, 133/126 te	96.6% te
Yang et al. [71]	Texture normalization w/ RBF-SVM	Snapshots	Unknown	N/A	N/A	5600/3600 tr, unknown te	97.2% te
		FERET	Frontal-only, studio			1400/900 tr, 3529 total te	92.2% te

8.5.1 Network Architecture

The network architectures and approach that we use are similar to the work of [60]. The input size of the image is 64×64. While training we used Adam [45] optimization method because it does not require hand-tuning of the learning rate, momentum, and other hyper-parameters.

For learning z we use the same hyper-parameters as learned in the previous model. In our experiments, running 2000 epochs helped converge to a low loss.

8.5.2 Database and Preprocessing

In order to train $\mathcal{G}_z$ and $\mathcal{D}_x$ our approach initially was to use the same input images as used to train $\mathcal{M}_k$. However, this resulted in the network not converging to a low loss. In other words, we were not able to learn a generative distribution that the generator could sample from. Our analysis and intuition suggested that in order for the adversarial loss to work, the task had to be a challenge for both $\mathcal{G}_z$ and $\mathcal{D}_x$. Our fully frontal pose images were ill-posed for this model.

Hence to train the GAN, we used the Labeled Faces in the Wild (LFW) database [47] and aligned the faces using dLib as provided by the OpenFace [1]. We trained on the entire dataset comprising around 13,000 images with 128 images held out for the purpose of qualitatively showing the image recovery results as in Fig. 8.11. In this case, by visual analysis as well by analytical analysis the $\mathcal{G}_z$ was better able to learn a distribution of z, p_g, that was a strong representation of the data, p_{data}. That is symbolically, $p_g = p_{\text{data}}$. The results and detailed evaluation of the model are done later in Sect. 8.5.3. In this part of the experiment, we use a subset of the PCSO database containing 10,000 images (5,000 male and 5,000 female) for testing the gender classification accuracy. The reason we did not test on the entire 400 K PCSO is simply because the occlusion removal step involves an iterative solver which is time consuming.

8.5.3 Experiment III: Further Occlusion Robustness via Deep Generative Approach

As shown in Fig. 8.11, we chose to model occlusions based on percentage of pixels missing from the center of the image. The shown images are from the held-out portion of the LFW dataset. We show recovered images in Fig. 8.11 as a visual confirmation that the DCGAN is able to recover unseen images with high fidelity, even under pretty heavy occlusions as much as 75% in the face center. The recovery results on the PCSO datasets to be tested for gender classification are comparable to that of the LFW, but we are not allowed to display mugshot images in any published work.

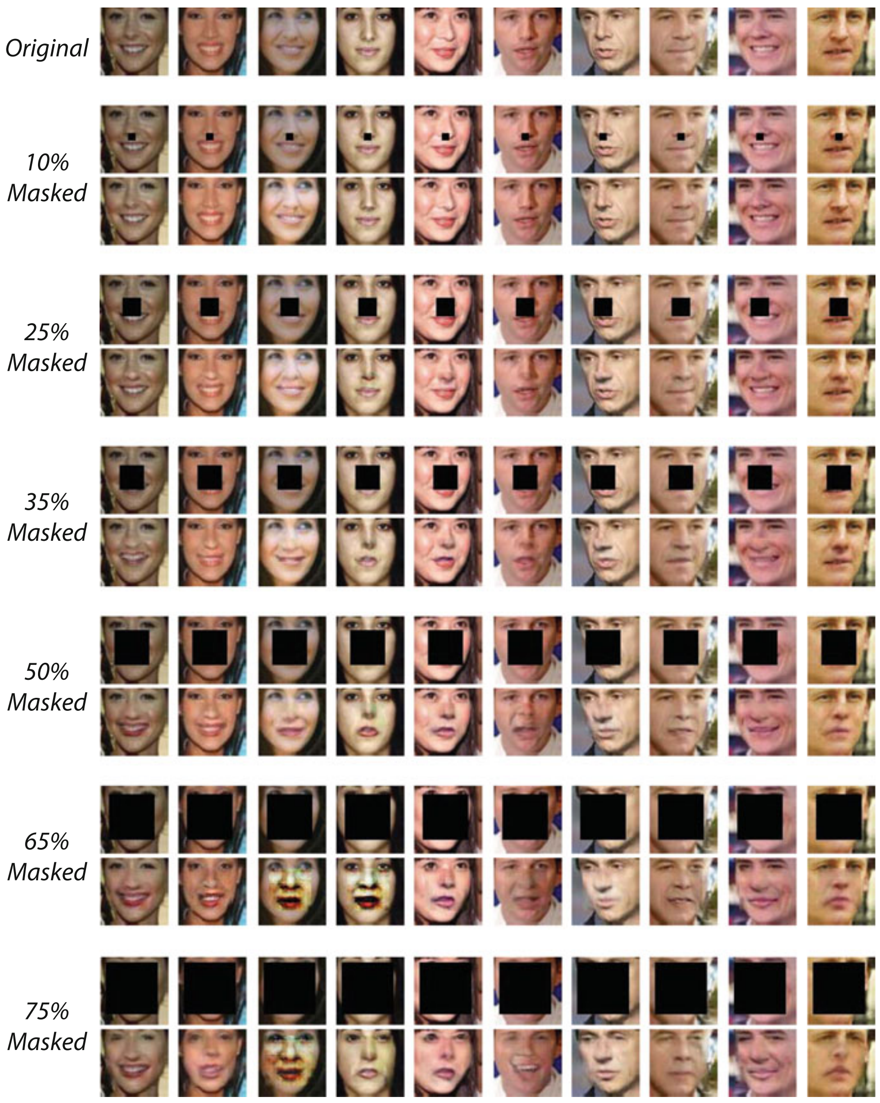


Fig. 8.11 Qualitative results of occlusion removal using DCGAN on images from LFW dataset

For training the $\mathcal{G}_z$ and $\mathcal{D}_x$, we used the LFW data that captures a high variance of poses, illumination, and faces. We found this was critical in helping especially the $\mathcal{G}_z$ converge to a stable weights. The model was able to generalize well to various faces and poses. As can be seen in Fig. 8.11, the model is able to generate missing information effectively.

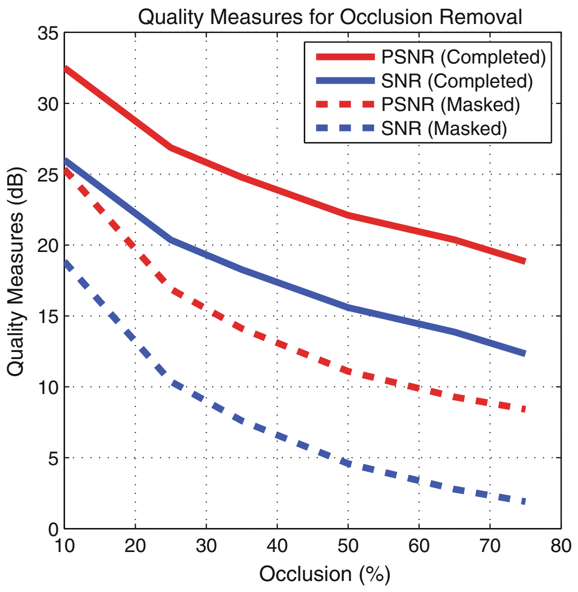


Fig. 8.12 Quality measure of occlusion removal on the PCSO subset

Table 8.9 Image quality measures (in dB) on the masked and occlusion removed images

Occlusion (%)	😷	😀	😷	😀
	PSNR	PSNR	SNR	SNR
10	25.3607	32.5194	18.8488	26.0101
25	16.8941	26.8720	10.3822	20.3627
35	14.112	24.7825	7.6001	18.2732
50	11.0937	22.1043	4.5819	15.5950
65	9.2849	20.3719	2.773	13.8627
75	8.4169	18.8392	1.905	12.3299

Not relying on visual inspection, we plotted the PSNR and SNR of the recovered (occlusion removed) faces from the 10K PCSO subset in Fig. 8.12. This is our quality measure of occlusion removal using GAN. As can be seen in Table 8.9, the PSNR/SNR is better for completed images and as expected is higher for images with lesser occlusions.

The primary motivation behind training a GAN was to improve the classification of $\mathcal{M}_k$ models. This is covered in Table 8.10. The first column of the table is our baseline case. This was constructed using upsampled images from the resolution as needed by $\mathcal{G}_z$ and $\mathcal{D}_x$ to the resolution expected by $\mathcal{M}_k$. All other accuracies should be evaluated with respect to this case. (Figure 8.13)

Table 8.10 Overall classification accuracy for Experiment III on the PCSO subset. Images are corrupted with **centered contiguous missing pixels** of various percentages

		😷	😀	😷	😀	😷	😀	😷	😀	😷	😀	😷	😀
Corrup.	0%	10%	10%	25%	25%	35%	35%	50%	50%	65%	65%	75%	75%
$\mathcal{M}_F$	96.64	85.39	96.15	71.12	95.29	56.08	93.60	54.05	87.88	57.82	78.42	50.19	78.56
$\mathcal{M}_1$	96.13	87.09	95.58	62.25	94.39	53.77	93.18	54.17	87.92	59.85	79.69	51.20	79.68
$\mathcal{M}_2$	95.74	85.73	94.75	68.30	93.46	53.59	92.47	55.03	86.45	51.26	78.19	50.13	78.81
$\mathcal{M}_3$	95.44	85.43	94.94	73.14	93.36	51.65	92.18	53.19	86.82	50.81	78.97	50.09	79.43
$\mathcal{M}_4$	97.52	83.35	96.97	56.34	94.78	50.24	92.82	53.58	86.93	52.72	78.96	50.12	79.70
$\mathcal{M}_5$	95.78	86.19	94.97	53.98	92.47	50.33	91.05	52.44	86.62	52.20	78.24	50.23	79.03
$\mathcal{M}_6$	96.21	86.41	95.59	58.47	92.84	50.42	91.30	52.02	87.15	51.02	78.74	50.40	79.77
$\mathcal{M}_P$	93.46	87.55	93.25	77.56	93.21	76.44	91.73	69.28	88.39	76.33	79.42	54.39	80.96

Figure 8.14 corresponds to the other columns of Table 8.10. The accuracies on completed images using $\mathcal{G}_z$ are significantly better than the accuracies on masked images. This suggests that the hallucinations are able to preserve the gender sensitivity of the original images.

The above statement can also be verified through visual inspection of Fig. 8.11. Even for high-percentage occlusions, $50 - 75\%$, the contours and features of the original face are pretty accurately reproduced by $\mathcal{G}_z$.

8.5.4 Discussion

In the current generative model for image occlusion removal, we assume that the occlusion mask is known to the algorithm, which is the **M** in Eq. 8.7. Although it is beyond the scope of this work to study how an algorithm can automatically determine the occlusion region, it will be an interesting research direction. For example, [61] is able to automatically tell which part is the face or the non-face region.

One big advantage of the DCGAN or GAN in general is that it is entirely unsupervised. The loss function is based off essentially measuring the similarity of two distributions (the true image distribution and the generated image distribution), rather than image-wise comparisons, which may require labeled ground-truth images be provided. The unsupervised nature of the GAN has made the training process much easier by not needing careful curating of the labeled data.

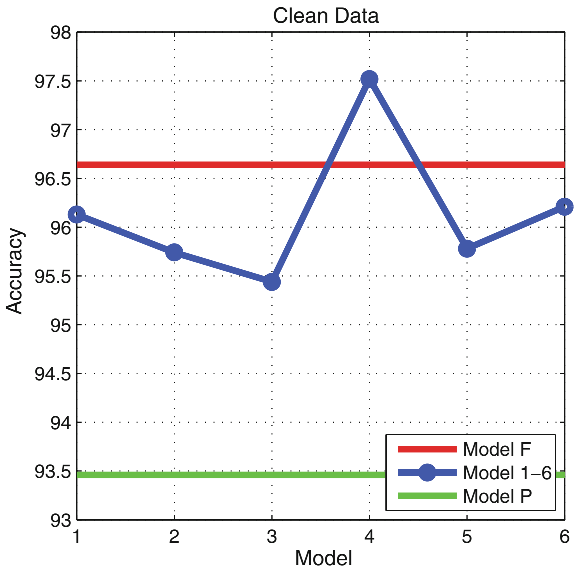


Fig. 8.13 Overall classification accuracy for Experiment III on the PCSO subset. Images are not corrupted

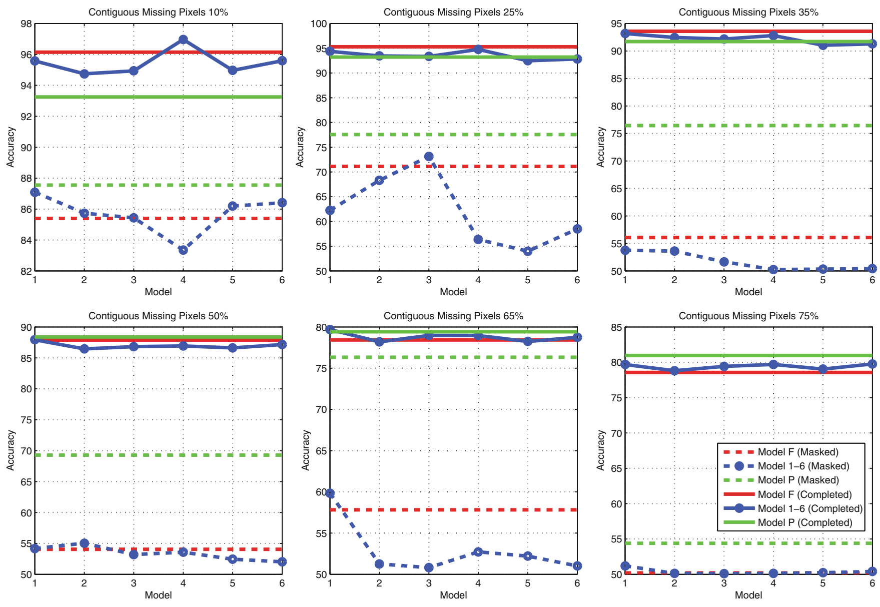


Fig. 8.14 Overall classification accuracy on the PCSO subset. Images are corrupted with **centered contiguous occlusions** of various percentages

The effectiveness of the DCGAN is also salient. It not only can recover the high-percentage missing data with high fidelity, which translates to significant improvement on the gender classification tasks, but also can be used for future data augmentation in a total unsupervised manner. We can essentially generate as much gender-specific data as needed, which will be an asset for training an even larger model.

During our experimentation, we find that training the DCGAN using more constrained faces (less pose variations, less lighting variations, less expression variations, etc.) such as the PCSO mugshot images actually degrades the recovery performance. The reason could be as follows. When the training data is less diversified, let us use one extreme case, which is a training dataset comprising thousands of images from only one subject. In this case, the 'true distribution' becomes a very densely clustered mass, which means whatever images the generator $\mathcal{G}$ tries to come up with, the discriminator $\mathcal{D}$ will (almost) always say that the generated ones are not from the true distribution, because the generated image distribution can hardly hit that densely clustered mass. This way, we are essentially giving a too easy task for the discriminator to solve, which prevents the discriminator to become a strong one, which leads to poorly performing generator as well in this adversarial setting. In a nutshell, during the DCGAN training, we definitely need more variations in the training corpus.

8.6 Conclusions and Future Work

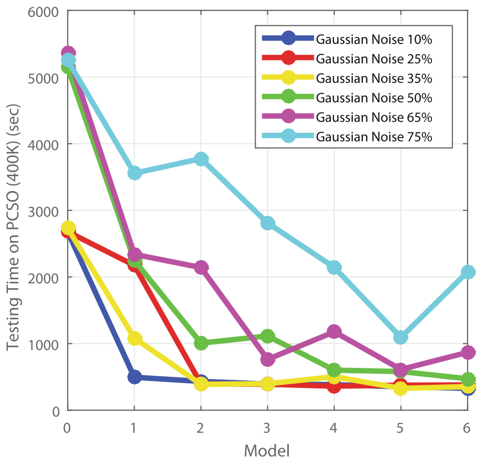
In this work, we have undertaken the task of occlusion and low-resolution robust facial gender classification. Inspired by the trainable attention model via deep architecture, and the fact that the periocular region is proven to be the most salient region for gender classification purposes, we are able to design a progressive convolutional neural network training paradigm to enforce the attention shift during the learning process. The hope is to enable the network to attend to particular high-profile regions (e.g., the periocular region) without the need to change the network architecture itself. The network benefits from this attention shift and becomes more robust toward occlusions and low-resolution degradations. With the progressively trained CNN models, we have achieved better gender classification results on the large-scale PCSO mugshot database with 400 K images under occlusion and low-resolution settings, compared to the one undergone traditional training. In addition, our progressively trained network is sufficiently generalized so that it can be robust to occlusions of arbitrary types and at arbitrary locations, as well as low resolution.

To further improve the gender classification performance on occluded facial images, we invoke a deep generative approach via deep convolutional generative adversarial networks (DCGAN) to fill in the missing pixels for the occluded facial regions. The recovered images not only show high fidelity as compared to the original un-occluded images but also significantly improve the gender classification performance.

In summary, on one hand, we aim at building a robust gender classifier that is tolerant to image degradations such as occlusions and low resolution, and on the other hand, we aim at mitigating and eliminating the degradations through a generative approach. Together, we are able to push the boundary of unconstrained facial gender classification.

Future work: We have carried out a set of large-scale testing experiments on the PCSO mugshot database with 400 K images, shown in the experimental section. We have noticed that, under the same testing environment, the amount of time it takes to test on the entire 400 K images various dramatically for different progressively trained models ($\mathcal{M}_0 - \mathcal{M}_6$). As shown in Fig. 8.15, we can observe a trend of testing time decrease when testing using $\mathcal{M}_0$ all the way to $\mathcal{M}_6$, where the curves correspond to the additive Gaussian noise occlusion robust experiments. This same trend is observed across the board for all the large-scale experiments on PCSO. The time difference is stunning. For example, if we look at the green curve, $\mathcal{M}_0$ takes over 5000 seconds while $\mathcal{M}_6$ only around 500. One of the future directions is to study the cause of this phenomenon. One possible direction is to study the sparsity or the smoothness of the learned filters.

Shown in our visualization (Fig. 8.15) of the 64 first-layer filters in AlexNet for models $\mathcal{M}_0$, $\mathcal{M}_3$, and $\mathcal{M}_6$, respectively, we can observe that the progressively trained filters seem to be smoother and this may be due to the implicit low-rank regularization phenomenon discussed in Sect. 8.3.1.3. Other future work may include studying how the ensemble of models [43, 44] can further improve the performance and how various



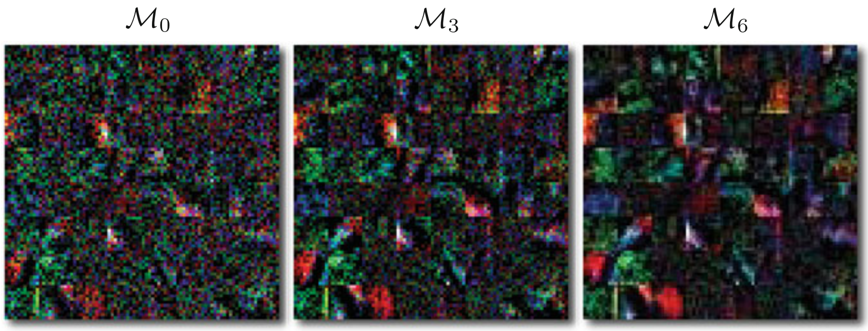


Fig. 8.15 (*Top*) Testing time for the additive Gaussian noise occlusion experiments on various models. (*Bottom*) Visualization of the 64 first-layer filters for models $\mathcal{M}_0$, $\mathcal{M}_3$, and $\mathcal{M}_6$, respectively

multimodal soft-biometrics traits [5, 23, 25, 26, 28–30, 37, 41, 42, 64, 66, 74] can be fused for improved gender classification, especially under more unconstrained scenarios.

References

1. B. Amos, B. Ludwiczuk, M. Satyanarayanan, Openface: a general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science (2016)
2. J. Ba, V. Mnih, K. Kavukcuoglu, Multiple object recognition with visual attention (2014), arXiv:1412.7755
3. D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate (2014), arXiv:1409.0473
4. A. Bartle, J. Zheng, Gender classification with deep learning. Stanford cs224d Course Project Report (2015)

5. P. Buchana, I. Cazan, M. Diaz-Granados, F. Juefei-Xu, M. Savvides, Simultaneous forgery identification and localization in paintings using advanced correlation filters, in *IEEE International Conference on Image Processing (ICIP)* (2016), pp. 1–5
6. W. Chan, N. Jaitly, Q.V. Le, O. Vinyals, Listen, attend and spell (2015), arXiv:1508.01211
7. D.Y. Chen, K.Y. Lin, Robust gender recognition for uncontrolled environment of real-life images. IEEE Trans. Consum. Electron. **56**(3), 1586–1592 (2010)
8. M.M. Cheng, Z. Zhang, W.Y. Lin, P. Torr, BING: binarized normed gradients for objectness estimation at 300fps, in *IEEE CVPR* (2014), pp. 3286–3293
9. J.K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, Y. Bengio, Attention-based models for speech recognition, in *Advances in Neural Information Processing Systems* (2015), pp. 577–585
10. W. Chuan-xu, L. Yun, L. Zuo-yong, Algorithm research of face image gender classification based on 2-D gabor wavelet transform and SVM, in *ISCSCT'08. International Symposium on Computer Science and Computational Technology, 2008*, vol. 1 (IEEE, 2008), pp. 312–315
11. B.A. Golomb, D.T. Lawrence, T.J. Sejnowski, Sexnet: a neural network identifies sex from human faces, in *NIPS*, vol. 1 (1990), p. 2
12. I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, Generative adversarial nets, in *Advances in Neural Information Processing Systems* (2014), pp. 2672–2680
13. S. Gutta, J.R. Huang, P. Jonathon, H. Wechsler, Mixture of experts for classification of gender, ethnic origin, and pose of human faces. IEEE Trans. Neural Netw. **11**(4), 948–960 (2000)
14. J. Harel, C. Koch, P. Perona, Graph-based visual saliency, in *NIPS* (2006), pp. 545–552
15. X. Hou, J. Harel, C. Koch, Image signature: highlighting sparse salient regions. IEEE TPAMI **34**(1), 194–201 (2012)
16. S.Y.D. Hu, B. Jou, A. Jaech, M. Savvides, Fusion of region-based representations for gender identification, in *IEEE/IAPR IJCB* (2011), pp. 1–7
17. L. Itti, C. Koch, E. Niebur et al., A model of saliency-based visual attention for rapid scene analysis. IEEE Trans. Pattern Anal. Mach. Intell. **20**(11), 1254–1259 (1998)
18. T. Jabid, M.H. Kabir, O. Chae, Gender classification using local directional pattern (LDP), in *IEEE ICPR* (IEEE, 2010), pp. 2162–2165
19. M. Jaderberg, K. Simonyan, A. Zisserman, et al., Spatial transformer networks, in *NIPS* (2015), pp. 2008–2016
20. F. Juefei-Xu, M. Savvides, Can your eyebrows tell me who you are?, in *2011 5th International Conference on Signal Processing and Communication Systems (ICSPCS)* (2011), pp. 1–8
21. F. Juefei-Xu, M. Savvides, Unconstrained periocular biometric acquisition and recognition using COTS PTZ camera for uncooperative and non-cooperative subjects, in *2012 IEEE Workshop on Applications of Computer Vision (WACV)* (2012), pp. 201–208
22. F. Juefei-Xu, M. Savvides, An augmented linear discriminant analysis approach for identifying identical twins with the aid of facial asymmetry features, in *2013 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2013), pp. 56–63
23. F. Juefei-Xu, M. Savvides, An image statistics approach towards efficient and robust refinement for landmarks on facial boundary, in *2013 IEEE Sixth International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2013), pp. 1–8
24. F. Juefei-Xu, M. Savvides, Subspace based discrete transform encoded local binary patterns representations for robust periocular matching on NIST's face recognition grand challenge. IEEE Trans. Image Process. **23**(8), 3490–3505 (2014)
25. F. Juefei-Xu, M. Savvides, Encoding and decoding local binary patterns for harsh face illumination normalization, in *IEEE International Conference on Image Processing (ICIP)* (2015), pp. 3220–3224
26. F. Juefei-Xu, M. Savvides, Facial ethnic appearance synthesis, in *Computer Vision - ECCV 2014 Workshops*. Lecture Notes in Computer Science, vol. 8926 (Springer International Publishing, Berlin, 2015), pp. 825–840
27. F. Juefei-Xu, M. Savvides, Pareto-optimal discriminant analysis, in *IEEE International Conference on Image Processing (ICIP)* (2015), pp. 611–615

28. F. Juefei-Xu, M. Savvides, Pokerface: partial order keeping and energy repressing method for extreme face illumination normalization, in *2015 IEEE Seventh International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2015), pp. 1–8
29. F. Juefei-Xu, M. Savvides, Single face image super-resolution via solo dictionary learning, in *IEEE International Conference on Image Processing (ICIP)* (2015), pp. 2239–2243
30. F. Juefei-Xu, M. Savvides, Weight-optimal local binary patterns, in *Computer Vision - ECCV 2014 Workshops*. Lecture Notes in Computer Science, vol. 8926 (Springer International Publishing, Berlin, 2015), pp. 148–159
31. F. Juefei-Xu, M. Savvides, Fastfood dictionary learning for periocular-based full face hallucination, in *2016 IEEE Seventh International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2016), pp. 1–8
32. F. Juefei-Xu, M. Savvides, learning to invert local binary patterns, in *27th British Machine Vision Conference (BMVC)* (2016)
33. F. Juefei-Xu, M. Savvides, Multi-class Fukunaga Koontz discriminant analysis for enhanced face recognition. Pattern Recognit. **52**, 186–205 (2016)
34. F. Juefei-Xu, M. Cha, J.L. Heyman, S. Venugopalan, R. Abiantun, M. Savvides, Robust local binary pattern feature sets for periocular biometric identification, in *4th IEEE International Conference on Biometrics: Theory Applications and Systems (BTAS)* (2010), pp. 1–8
35. F. Juefei-Xu, M. Cha, M. Savvides, S. Bedros, J. Trojanova, Robust periocular biometric recognition using multi-level fusion of various local feature extraction techniques, in *IEEE 17th International Conference on Digital Signal Processing (DSP)* (2011), pp. 1–7
36. F. Juefei-Xu, K. Luu, M. Savvides, T. Bui, C. Suen, Investigating age invariant face recognition based on periocular biometrics, in *2011 International Joint Conference on Biometrics (IJCB)* (2011), pp. 1–7
37. F. Juefei-Xu, C. Bhagavatula, A. Jaech, U. Prasad, M. Savvides, Gait-ID on the move: pace independent human identification using cell phone accelerometer dynamics, in *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)* (2012), pp. 8–15
38. F. Juefei-Xu, D.K. Pal, M. Savvides, Hallucinating the full face from the periocular region via dimensionally weighted K-SVD, in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2014), pp. 1–8
39. F. Juefei-Xu, D.K. Pal, M. Savvides, Methods and software for hallucinating facial features by prioritizing reconstruction errors (2014). U.S. Provisional Patent Application Serial No. 61/998,043, 17 Jun 2014
40. F. Juefei-Xu, K. Luu, M. Savvides, Spartans: single-sample periocular-based alignment-robust recognition technique applied to non-frontal scenarios. IEEE Trans. Image Process. **24**(12), 4780–4795 (2015)
41. F. Juefei-Xu, D.K. Pal, M. Savvides, NIR-VIS heterogeneous face recognition via cross-spectral joint dictionary learning and reconstruction, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2015), pp. 141–150
42. F. Juefei-Xu, D.K. Pal, K. Singh, M. Savvides, A preliminary investigation on the sensitivity of COTS face recognition systems to forensic analyst-style face processing for occlusions, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2015), pp. 25–33
43. F. Juefei-Xu, V.N. Boddeti, M. Savvides, Local binary convolutional neural networks (2016), arXiv:1608.06049
44. F. Juefei-Xu, V.N. Boddeti, M. Savvides, Local binary convolutional neural networks, in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017)
45. D. Kingma, J. Ba, Adam: a method for stochastic optimization (2014), arXiv:1412.6980
46. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *NIPS* (2012), pp. 1097–1105
47. G.B.H.E. Learned-Miller, Labeled faces in the wild: updates and new reporting procedures. Technical Report UM-CS-2014-003, University of Massachusetts, Amherst (2014)

48. P.H. Lee, J.Y. Hung, Y.P. Hung, Automatic gender recognition using fusion of facial strips, in *IEEE ICPR* (IEEE, 2010), pp. 1140–1143
49. X. Leng, Y. Wang, Improving generalization for gender classification, in *IEEE ICIP* (IEEE, 2008), pp. 1656–1659
50. G. Levi, T. Hassner, Age and gender classification using convolutional neural networks, in *IEEE CVPRW* (2015)
51. H. Lin, H. Lu, L. Zhang, A new automatic recognition system of gender, age and ethnicity, in *WCICA 2006, The Sixth World Congress on Intelligent Control and Automation, 2006*, vol. 2 (IEEE, 2006), pp. 9988–9991
52. H. Lu, H. Lin, Gender recognition using adaboosted feature, in *ICNC 2007, Third International Conference on Natural Computation, 2007*, vol. 2 (IEEE, 2007), pp. 646–650
53. L. Lu, Z. Xu, P. Shi, Gender classification of facial images based on multiple facial regions, in *2009 WRI World Congress on Computer Science and Information Engineering*, vol. 6 (IEEE, 2009), pp. 48–52
54. M.T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, in *Conference on Empirical Methods in Natural Language Processing (EMNLP)* (2015)
55. A. Martinez, R. Benavente, The AR Face Database. CVC Technical report No. 24 (1998)
56. J. Merkow, B. Jou, M. Savvides, An exploration of gender identification using only the periocular region, in *IEEE BTAS* (2010), pp. 1–5
57. V. Mnih, N. Heess, A. Graves, et al., Recurrent models of visual attention, in *NIPS* (2014), pp. 2204–2212
58. B. Moghaddam, M.H. Yang, Gender classification with support vector machines, in *IEEE FG* (IEEE, 2000), pp. 306–311
59. D.K. Pal, F. Juefei-Xu, M. Savvides, Discriminative invariant kernel features: a bells-and-whistles-free approach to unsupervised face recognition and pose estimation in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016)
60. A. Radford, L. Metz, S. Chintala, Unsupervised representation learning with deep convolutional generative adversarial networks (2015), arXiv:1511.06434
61. S. Saito, T. Li, H. Li, Real-time facial segmentation and performance capture from RGB input (2016), arXiv:1604.02647
62. M. Savvides, F. Juefei-Xu, Image Matching Using Subspace-Based Discrete Transform Encoded Local Binary Patterns (2013). US Patent US 2014/0212044 A1
63. M. Savvides, F. Juefei-Xu, U. Prabhu, C. Bhagavatula, Unconstrained biometric identification in real world environments, in *Advances in Human Factors and System Interactions*. Advances in Intelligent Systems and Computing, vol. 497 (2016), pp. 231–244
64. K. Seshadri, F. Juefei-Xu, D.K. Pal, M. Savvides, Driver cell phone usage detection on strategic highway research program (SHRP2) face view videos, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2015), pp. 35–43
65. C. Tai, T. Xiao, X. Wang, W. E, Convolutional neural networks with low-rank regularization. ICLR (2016), http://arxiv.org/abs/1511.06067
66. S. Venugopalan, F. Juefei-Xu, B. Cowley, M. Savvides, Electromyograph and keystroke dynamics for spoof-resistant biometric authentication, in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2015), pp. 109–118
67. O. Vinyals, A. Toshev, S. Bengio, D. Erhan, Show and tell: a neural image caption generator, in *IEEE CVPR* (2015), pp. 3156–3164
68. T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, Z. Zhang, The application of two-level attention models in deep convolutional neural network for fine-grained image classification, in *IEEE CVPR* (2015), pp. 842–850
69. H. Xu, K. Saenko, Ask, attend and answer: exploring question-guided spatial attention for visual question answering (2015), arXiv:1511.05234
70. K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, Y. Bengio, Show, attend and tell - neural image caption generation with visual attention, in *ICML* (2015), pp. 2048–2057

71. Z. Yang, M. Li, H. Ai, An experimental study on automatic face gender classification, in *IEEE ICPR*, vol. 3 (IEEE, 2006), pp. 1099–1102
72. L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, A. Courville, Describing videos by exploiting temporal structure, in *IEEE ICCV* (2015), pp. 4507–4515
73. R. Yeh, C. Chen, T.Y. Lim, M. Hasegawa-Johnson, M.N. Do, Semantic image inpainting with perceptual and contextual losses (2016), arXiv:1607.07539
74. N. Zehngut, F. Juefei-Xu, R. Bardia, D.K. Pal, C. Bhagavatula, M. Savvides, Investigating the feasibility of image-based nose biometrics, in *IEEE International Conference on Image Processing (ICIP)* (2015), pp. 522–526
75. Y. Zhu, O. Groth, M. Bernstein, L. Fei-Fei, Visual7W: grounded question answering in images (2015), arXiv:1511.03416

Chapter 9
Gender Classification from NIR Iris Images Using Deep Learning

Juan Tapia and Carlos Aravena

Abstract Gender classification from NIR iris image is a new topic with only a few papers published. All previous work on gender-from-iris tried to find the best feature extraction techniques to represent the information of the iris texture for gender classification using normalized, encoded or periocular images. However this is a new topic in deep-learning application with soft biometric. In this chapter, we show that learning gender-iris representations through the use of deep neural networks may increase the performance obtained on these tasks. To this end, we propose the application of deep-learning methods to separate the gender-from-iris images even when the amount of learning data is limited, using an unsupervised stage with Restricted Boltzmann Machine (RBM) and a supervised stage using a Convolutional Neural Network (CNN).

9.1 Introduction

One active area in 'soft biometrics' research involves classifying the gender of the person from a biometric sample. Most work done on gender classification has involved the analysis of faces or periocular images. Various types of classifiers have been used in gender classification after feature extraction and selection. The classifiers that have yielded highest classification accuracy in gender-from-face are Adaboost, RBF networks, and Support vector machines (SVM) [1, 20, 22, 24, 25, 31]. Recently Levi et al. [19] proposed an automatic face age and gender classification using a simple convolutional net architecture that can be used when the quantity of learning data

J. Tapia (✉)
Universidad Andres Bello, Av. Antonio Varas 880, Santiago, Chile
e-mail: juan.tapia@unab.cl

C. Aravena
Universidad de Chile, Av. Tupper 2007, Santiago, Chile
e-mail: cararave@ing.uchile.cl

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_9

is limited. Also Juefei-Xu et al. [15] classified gender using faces images with occlusion and low resolution using a progressive trained convolutional neural network with attention.

Gender classification using iris information is a rather new topic, with only a few papers published [2, 17, 26, 28]. Most gender classification methods reported in the literature use all iris texture features for classification. As a result, gender-irrelevant information might be fed into the classifier which may result in poor generalization, especially when the training set is small. It has been shown both theoretically and empirically that reducing the number of irrelevant or redundant features increases the learning efficiency of the classifier [21].

In terms of comparing iris codes for identity recognition, iris codes of different individuals, and even of the left and right eyes of the same individual, have been shown to be independent. At the same time, several authors have reported that, using an analysis of iris texture different from that used for identity recognition, it is possible to classify the gender of the person with an accuracy much higher than chance.

There are several reasons why gender-from-iris is an interesting and potentially useful problem [27]. One possible use arises in searching an enrolled database for a match. If the gender of the sample can be determined, then it can be used to order the search and reduce the average search time. Another possible use arises in social settings where it may be useful to screen entry to some area based on gender, but without recording identity. Gender classification is also important for demographic information collection, marketing research, and real-time electronic marketing. Another possible use is in high-security scenarios, where there may be value in knowing the gender of the people who attempt entry but are not recognized as any of the enrolled persons. And, at a basic science level, it is of value to more fully understand what information about a person can be extracted from analysis of their iris texture.

Thomas et al. [28] were the first to explore gender-from-iris, using images acquired with an LG 2200 sensor. They segmented the iris region and employed machine learning techniques to develop models that predict gender based on the iris texture features. They segmented the iris region, created a normalized iris image, and then applied a log-Gabor filter to the normalized image. They ran a rigorous quality check to discard images with poor quality such as motion blur or out of focus. In addition to the log-Gabor texture features, they used seven geometric features of the pupil and iris, and were able to reach an accuracy close to 80%.

Lagree et al. [17] experimented with iris images acquired using an LG 4000 sensor. They computed texture features separately for eight five-pixel horizontal bands, running from the pupil-iris boundary out to the iris sclera boundary, and ten 24-pixel vertical bands from a 40×240 image. The normalized image is not processed by the log-Gabor filters that are used by IrisBEE software [23] to create the 'iris code' for biometrics purpose and do not use any geometrics features to develop models that predict gender and ethnicity based on the iris texture feature. These are the differences from features computed by Thomas in [28]. This approach reached an accuracy close to 62% for gender and close to 80% for ethnicity.

Bansal et al. [2] experimented with iris images acquired with a Cross Match SCAN-2 dual-iris camera. An statistical feature extraction technique based on correlation between adjacent pixels was combined with a 2D wavelet tree based on feature extraction techniques to extract significant features from the iris image. This approach reached an accuracy of 83.06% for gender classification. Nevertheless, the database used in this experiment was very small (300 images) compared to other studies published in the literature.

Tapia et al. [26] experimented with iris images acquired using an LG 4000 sensor to classify gender of a person based on analysis of features of the iris texture. They used different implementations of Local Binary Patterns from the iris image using the masked information. Uniform LBP with concatenated histograms significantly improves accuracy of gender prediction relative to using the whole iris image. Using a non subject-disjoint test set, they were able to achieve over 91% correct gender prediction using the texture of the left iris.

Costa-Abreu et al. [8] explored the gender prediction task with respect to three approaches using only geometric features, only texture features and both geometric and texture features extracted from iris images. This work used a BioSecure Multimodal DataBase (BMDB) and these images were taken using a LG Iris Access EOU-3000. They were able to achieve over 89.74% correct gender prediction using the texture of the iris. Nevertheless the dataset is not available and the author chose the images, spectacles were not allowed to be worn by subjects, although contact lenses were allowed. Our database is a more real representation.

Bobeldik et al. [4] explored the gender prediction accuracy associated with four different regions from NIR iris images: the extended ocular region, the iris-excluded ocular region, the iris-only region, and the normalized iris-only region. They used a Binarized Statistical Image Feature (BSIF) texture operator [16] to extract features from the regions previously defined. The ocular region reached the best performance with 85.7% while normalized images exhibited the worst performance, with almost a 20% difference in performance over the ocular region (65%). Thereby we can understand that the normalization process may be filtering out useful information.

Recently, Tapia et al. [27] predicted gender directly from the same binary iris code that could be used for recognition. They found that information for gender prediction is distributed across the iris, rather than localized in particular concentric bands. They also found that using selected features representing a subset of the iris region achieves better accuracy than using features representing the whole iris region achieving 89% correct gender prediction using the fusion of the best features of iris code from the left and the right eyes.

A summary of these methods' experimental setup is presented in Table 9.1.

9.2 Deep-Learning Models

In this chapter, we first propose an unsupervised approach to use a big quantity of unlabeled iris images to pretrain a Deep Belief Network [13] and then use it to make a fast fine tuning of a Deep Multilayer Network (MLP) for gender classification.

Table 9.1 Gender classification summary of previously published papers. N represents: Normalized Image, E represents: Encoded Image, P represent Periocular Image

Paper	Sensor	Size of images	Number of images	Number of subject	TI	E
Thomas et al. [28]	LG 2200	20 × 240	16,469	N/A	N	–
Lagree et al. [17]	LG 4000	40 × 240	600	300	N	–
Bansal et al. [2]	Cross Match SCAN-2	10 × 500	300	200	N	–
Tapia et al. [26]	LG 4000	20 × 240	1,500	1,500	N	–
Costa-Abreu et al. [8]	LG EOU-3000	20 × 240	1,600	200	N	–
Bobeldyk et al. [4]	NIR sensor	20 × 240	3,314	1,083	N/P	–
Tapia et al. [27]	LG 4000	20 × 240	3,000	1,500	–	E

A second approach is proposed using a Lenet-5 [18] Convolutional Neural Network (CNN) model to improve the gender classification from normalized Near-Infrared Red (NIR) iris images. We then compared and discuss the results obtained by both methods.

9.2.1 *Semi-supervised Method*

In the iris biometric scenario we usually have a lot of information without labels (age, gender, ethnicity), because iris databases were created focusing on iris identification with encoding or periocular images and not in soft biometric problems such as classifying gender-from-iris. Thus, we commonly have databases with iris images taken for the same subject across several sessions and many of them do not have the gender information available. Because of this, it can be a hard task to create a large person-disjoint dataset to proper train and evaluate soft biometrics iris algorithms. DBN unsupervised algorithm may help to classify unlabeled iris images.

9.2.1.1 Deep Belief Network

Deep Belief Network (DBN) consists of multiple layers of stochastic latent variables trained using an unsupervised learning algorithm followed by a supervised learning phase using feed-forward back-propagation Multilayer Neural Networks (MLP) [13]. In the unsupervised pretraining stage, each layer is trained using a Restricted Boltzmann Machine (RBM). Unsupervised pretraining is an important step in solving a classification problem with terabytes of data and high variability without labels that is the main concern in this proposal. A DBN is a graphical model where neurons of the hidden layer are conditionally independent of one another for a particular configuration of the visible layer and viceversa. A DBN can be trained layer-wise by iteratively maximizing the conditional probability of the input vectors or visible vectors given the hidden vectors and a particular set of layer weights. This layer-wise training can help in improving the variational lower bound on the probability of the input training data, which in turn leads to an improvement of the overall generative model. DBNs are graphical models which learn to extract a deep hierarchical representation of the training data. They model the joint distribution between observed vector x and the l hidden layers h^k as follows:

$$P(x, h^k, \ldots, h^l) = \left(\prod_{k=0}^{l-2} P(h^k | h^{k+1}) \right) P(h^{l-1} - 1, h^l) \tag{9.1}$$

where $x = h^0$, $P(h^{k-1}|, h^k)$ is a conditional distribution for the visible units conditioned on the hidden units of the RBM at level k, and $P(h^{l-1}, h^l)$ is the visible-hidden joint distribution in the top-level RBM. This is illustrated in Fig. 9.1.

In according with Hinton et al. [13] The principle of greedy layer-wise unsupervised training can be applied to DBNs with RBMs as the building blocks for each layer [3]. The process is as follows:

1. Train the first layer as an RBM that models the raw input $x = h(0)$ as its visible layer.
2. Use that first layer to obtain a representation of the input that will be used as data for the second layer. Two common solutions exist. This representation can be chosen as being the mean activations $p(h(1) = 1|h(0))$ or samples of $p(h(1)|h(0))$.
3. Train the second layer as an RBM, taking the transformed data (samples or mean activations) as training examples (for the visible layer of that RBM).
4. Iterate (2 and 3) for the desired number of layers, each time propagating upward either samples or mean values.
5. Fine tune all the parameters of this deep architecture with respect to a proxy for the DBN log-likelihood, or with respect to a supervised training criterion (after adding extra learning machinery to convert the learned representation into supervised predictions).

The RBM can be denoted by the energy function E

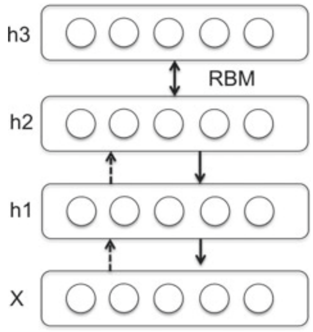


Fig. 9.1 Block diagram of RBM. X represent input layer, h1–h3 hidden layer

$$E(v,h) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_i h_j w_{i,j} v_i \tag{9.2}$$

where, the RBM consists of a matrix of layer weights $W = (w_{i,j})$ between the hidden units h_j and the visible units v_j. The a_i and b_j are the bias weights for the visible units and the hidden units respectively. The RBM takes the structure of a bipartite graph and hence it only has inter-layer connections between the hidden or visible layer neurons but no intra-layer connections within the hidden or visible layers. So, the activations of the visible unit neurons are mutually independent for a given set of hidden unit activations and viceversa. Hence, by setting either h or v constant, we can compute the conditional distribution of the other as follows [13]:

$$P(h_j = 1 \mid v) = \sigma\left(b_j + \sum_{i=1}^{m} w_{i,j} v_i\right) \tag{9.3}$$

$$P(v_i = 1 \mid h) = \sigma\left(a_i + \sum_{j=1}^{n} w_{i,j} h_i\right) \tag{9.4}$$

where σ denotes the log sigmoid function. The training algorithm maximizes the expected log probability assigned to the training dataset V. So if the training dataset V consists of the visible vectors v, then the objective function is as follows:

$$argmax E\left[\sum_{v \in V} log P(v)\right] \tag{9.5}$$

A RBM is trained using a Contrastive Divergence algorithm [7]. Once trained, the DBN can be used to initialize the weights of the MLP for the supervised learning phase.

In this work, we also focus on fine tuning via supervised gradient descent (Fig. 9.2). Specifically, we use a logistic regression classifier to classify the input x based on the output of the last hidden layer $h(l)$ of the DBN. Fine tuning is then performed via supervised gradient descent of the negative log-likelihood cost function. Since the supervised gradient is only non-null for the weights and hidden layer biases of each layer (i.e., null for the visible biases of each RBM), this procedure is equivalent to initializing the parameters of a deep MLP with the weights and hidden layer biases obtained with the unsupervised training strategy. One can also observe that the DBN is very similar with the one for Stack denoise Autoencoding [30], because both involve the principle of unsupervised layer-wise pre-training followed by supervised fine tuning as a Deep MLP. In this chapter we use the RBM class instead of the denoise autocoding class. Now the main problem is to choose the best hyperparameters for Deep MLP.

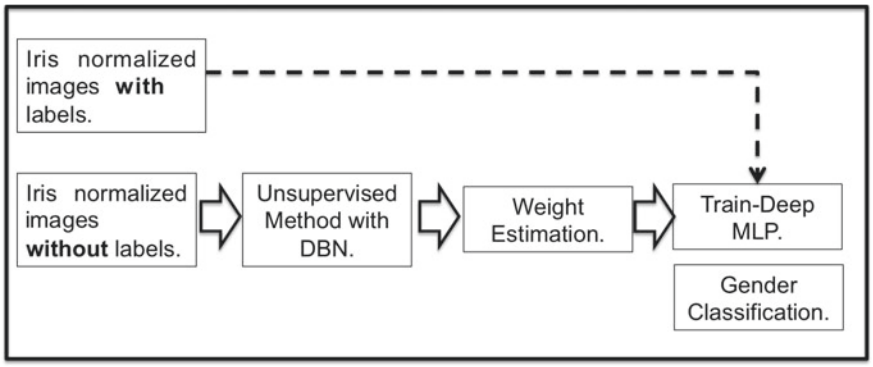

Iris normalized images **with** labels.
Iris normalized images **without** labels.
Unsupervised Method with DBN.
Weight Estimation.
Train-Deep MLP.
Gender Classification.

Fig. 9.2 Block diagram with the main stages from semi-supervised iris gender classification. Pretrained (Unsupervised Stage) was made with DBN and fine tuning with Deep MLP (Supervised Stage)

Once all layers are pretrained, the network goes through a second stage of training called fine tuning. Here we consider supervised fine tuning where we want to minimize prediction error on a supervised task. For this, we first add a logistic regression layer on top of the network (more precisely on the output code of the output layer). Then train the entire network as we would train a Deep MLP. This stage is supervised, since now we use the target class during training.

9.2.2 Supervised Method

A second approach is proposed using a Lenet-5 [20] Convolutional Neural Network (CNN) model to improve gender classification from Normalized Near Infrared Red (NIR) iris images.

9.2.2.1 CNN

Convolutional Neural Networks (CNN) are biologically inspired variants of MLPs [12]. From Hubel and Wiesel's early work on the cat's visual cortex [14], we know the visual cortex contains a complex arrangement of cells. These cells are sensitive to small subregions of the visual field, called a receptive field. The subregions are tiled to cover the entire visual field. These cells act as local filters over the input space and are well-suited to exploit the strong spatially local correlation present in natural images. Additionally, two basic cell types have been identified: Simple cells respond maximally to specific edge-like patterns within their receptive field. Complex cells have larger receptive fields and are locally invariant to the exact position of the pattern.

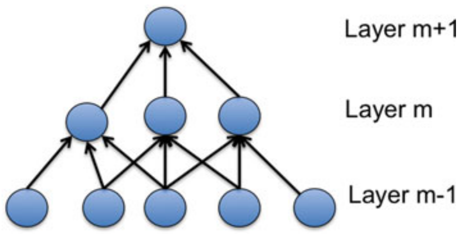


Fig. 9.3 Representation of sparse connectivity

9.2.2.2 Sparse Connectivity

CNNs exploit spatially local correlation by enforcing a local connectivity pattern between neurons of adjacent layers. In other words, the inputs of hidden units in layer m are from a subset of units in layer $m-1$, units that have spatially contiguous receptive fields. We can illustrate this graphically as follows: Imagine that layer $m-1$ is the input retina. In Fig. 9.3, units in layer m have receptive fields of width 3 in the input retina and are thus only connected to three adjacent neurons in the retina layer. Units in layer $m+1$ have a similar connectivity with the layer below. We say that their receptive field with respect to the layer below is also 3, but their receptive field with respect to the input is larger (5). Each unit is unresponsive to variations outside of its receptive field with respect to the retina. The architecture thus ensures that the learnt 'filters' produce the strongest response to a spatially local input pattern.

However, as show in Fig. 9.3 stacking many such layers leads to (nonlinear) 'filters' that become increasingly 'global' (i.e., responsive to a larger region of pixel space). For example, the unit in hidden layer $m+1$ can encode a nonlinear feature of width 5 (in terms of pixel space). One of the first applications of convolutional neural networks (CNN) is perhaps the LeNet-5 network described by [18] for optical character recognition. Compared to modern deep CNN, their network was relatively modest due to the limited computational resources of the time and the algorithmic challenges of training bigger networks. Though much potential laid in deeper CNN architectures (networks with more neuron layers), only recently have they became prevalent, following the dramatic increase in both the computational power, due to availability of Graphical Processing, Units (GPU); the amount of training data readily available on the Internet; and the development of more effective methods for training such complex models. One recent and notable examples is the use of deep CNN for image classification on the challenging Imagenet benchmark [11]. Deep CNN have additionally been successfully applied to applications including human pose estimation, facial keypoint detection, speech recognition, and action classification. To our knowledge, this is the first report of their application to the task of gender classification from iris images.

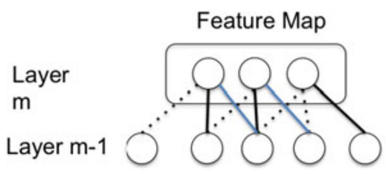


Fig. 9.4 Representation of shared weight on CNN

9.2.2.3 Shared Weight

In addition, in CNNs, each filter h_i is replicated across the entire visual field. These replicated units share the same parametrization (weight vector and bias) and form a feature map. In Fig. 9.4, we show three hidden units belonging to the same feature map. Weights of the same color are shared and constrained to be identical. Gradient descent can still be used to learn such shared parameters, with only a small change to the original algorithm. The gradient of a shared weight is simply the sum of the gradients of the parameters being shared. Replicating units in this way allows for features to be detected regardless of their position in the visual field. Additionally, weight sharing increases learning efficiency by greatly reducing the number of free parameters being learnt. The constraints on the model enable CNNs to achieve better generalization on vision problems.

9.2.2.4 Pooling

Another important concept of CNNs are the pooling techniques. Today one of the most used pooling algorithms is Max Pooling, but others can also be used such as: Average, Sum, and Mean Pooling. The pool layers are used mainly to down-sample the volumes spatially and to reduce the maps of features of previous layers. This chapter uses a Max pooling approach.

Max pooling partitions the input image into a set of non-overlapping rectangles (subregions) and, for each subregion, outputs the maximum value. Max pooling is useful in for two main reasons: (a) by eliminating non-maximal values, it reduces computation for upper layers; and (b) it provides a form of translation invariance. Imagine cascading a max pooling layer with a convolutional layer. There are eight directions in which one can translate the input image by a single pixel. If max pooling is done over a 2×2 region, three out of these eight possible configurations will produce exactly the same output at the convolutional layer. When max pooling is applied over a 3×3 window, this jumps to $5/8$. Since it provides additional robustness to position, max pooling is a 'smart' way of reducing the dimensionality of intermediate representations within a convolutional neural network.

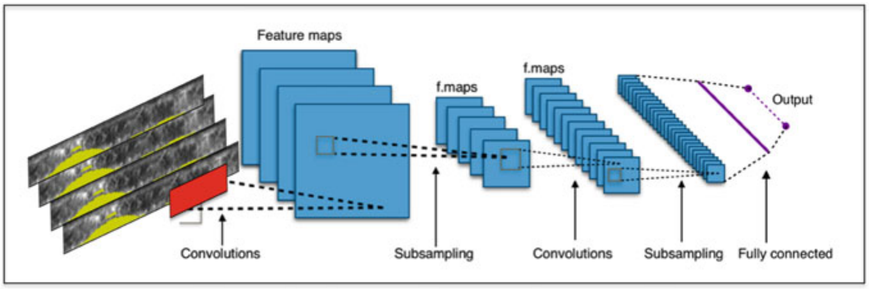


Fig. 9.5 Representation with the main stages from Lenet-5 CNN for iris gender classification. The input are normalized NIR iris image with mask in *yellow*

9.2.2.5 Network Architecture

In this work we trained two Lenet-5 networks, named CNN-1 and CNN-2. Our proposed network architecture is used throughout our experiments for gender classification. The network comprises only three convolutional layers and two fully connected layers with a small number of neurons. Our choice of a smaller network design is motivated both from our desire to reduce the risk of overfitting as well as the nature of the problem we are attempting to solve (gender classification has only two classes). Only one channel (grayscale images with the occlusion mask) is processed directly by the network. Normalized iris images of sizes 20×240 are fed into the network.

Sparse, convolutional layers and max pooling are at the heart of the LeNet family of models. Figure 9.5 shows a graphical depiction of a LeNet model. The three subsequent convolutional layers are then defined as follows for CNN-1:

1. 20 filters of size $1 \times 5 \times 5$ pixels are applied to the input in the first convolutional layer, followed by a rectified linear operator (ReLU), a max pooling layer taking the maximal value of 2×2 regions with two-pixel strides and a local response normalization layer.
2. 50 filters of size $1 \times 10 \times 10$ pixels are applied to the input in the second convolutional layer, followed by a rectified linear operator (ReLU), a max pooling layer taking the maximal value of 2×2 regions with two-pixel strides and a local response normalization layer.
3. 100 filters of size $1 \times 20 \times 20$ pixels are applied to the input in the third convolutional layer, followed by a rectified linear operator (ReLU), a max pooling layer taking the maximal value of 2×2 regions with two-pixel strides and a local response normalization layer.

The following fully connected layers are then defined by:

4. A first fully connected layer that receives the output of the third convolutional layer and contains 4,800 neurons, followed by a ReLU and a dropout layer (0.2).

5. A second fully connected layer that receives the 3,840 dimensional output of the first fully connected layer and again contains 500 neurons, followed by a ReLU and a dropout layer (0.2).
6. Finally, the output of the last fully connected layer is fed to a softmax layer that assigns a probability for each class. The prediction itself is made by taking the class with the maximal probability for the given test image.

The first CNN was modified in order to use more appropriate filter sizes, considering that our images are wider than taller. The three subsequent convolutional layers are then defined as follows for CNN-2:

1. 128 filters of size $1 \times 5 \times 21$ pixels are applied to the input in the first convolutional layer, followed by a rectified linear operator (ReLU), a max pooling layer taking the maximal value of 2×2 regions with two-pixel strides and a local response normalization layer.
2. 256 filters of size $1 \times 5 \times 21$ pixels are applied to the input in the second convolutional layer, followed by a rectified linear operator (ReLU), a max pooling layer taking the maximal value of 2×2 regions with two-pixel strides and a local response normalization layer.
3. 512 filters of size $1 \times 2 \times 45$ pixels are applied to the input in the third convolutional layer, followed by a rectified linear operator (ReLU), a max pooling layer taking the maximal value of 2×2 regions with two-pixel strides and a local response normalization layer.
   The following fully connected layers are then defined by
4. A first fully connected layer that receives the output of the third convolutional layer and contains 4,800 neurons, followed by a ReLU and a dropout layer (0.5).
5. A second fully connected layer that receives the 2,400 dimensional output of the first fully connected layer and again contains 4,800 neurons, followed by a ReLU and a dropout layer (0.5).
6. Finally, the output of the last fully connected layer is fed to a softmax layer that assigns a probability for each class. The prediction itself is made by taking the class with the maximal probability for the given test image

9.3 Iris Normalized Images

In this work, we used normalized iris images because the main idea is to extract the information from the same process used for iris identification so gender classification could be used as a secondary stage of information.

The iris feature extraction process involves the following steps: First, a camera acquires an image of the eye. All commercial iris recognition systems use near-infrared illumination, to be able to image iris texture of both ‘dark’ and ‘light’ eyes [26]. Next, the iris region is located within the image. The annular region of the iris is transformed from raw image coordinates to normalized polar coordinates. This

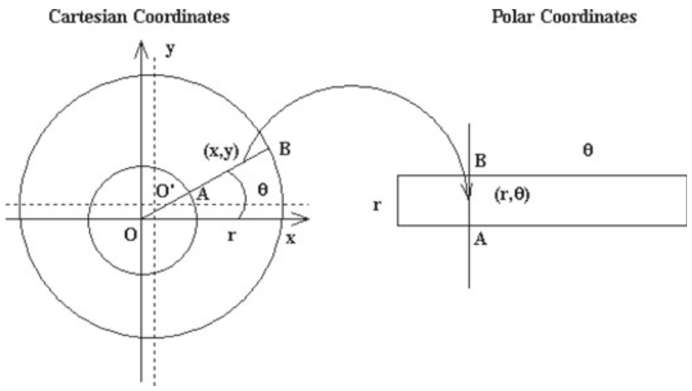


Fig. 9.6 Transformation of Cartesian coordinates (x, y) to Polar coordinates in order to generate the normalized image

results in what is sometimes called an 'unwrapped' or 'rectangular' iris image. See Fig. 9.6. A texture filter is applied at a grid of locations on this unwrapped iris image, and the filter responses are quantized to yield a binary iris code [5]. Iris recognition systems operating on these principles are widely used in a variety of applications around the world [6, 10, 29].

The radial resolution (r) and angular resolution (θ) used during the normalization or unwrapping step determine the size of the normalized iris image, and can significantly influence the iris recognition rate. This unwrapping is referred to as using Daugman's rubber sheet model [9]. In this work, we use a normalized image of $20(r) \times 240(\theta)$, created using Daugman's method and Osiris implementation. Both implementations also create a segmentation mask of the same size as the normalized image. The segmentation mask indicates the portions of the normalized iris image that are not valid due to occlusion by eyelids, eyelashes or specular reflections.

9.4 Dataset

The images used in this paper were taken with an LG 4000 sensor (labeled images) and IrisGuard AD-100 (unlabeled images). The LG 4000 and AD-100 use near-infrared illumination and acquire a 480×640, 8-bit/pixel image. Example of iris images appear in Fig. 9.7. We used two subsets, one for the unsupervised stage and the other for the supervised stage. The image dataset for the supervised stage is person-disjoint.

The whole database consists of two datasets: The first dataset has 10,000 unlabeled images, 5 K left iris images and 5 K right iris images. We do not know the quantities of male and female iris images. We used a fusion of 10 K (left and right) for the pretrained (unsupervised) stage. Currently this dataset is not publicly available.

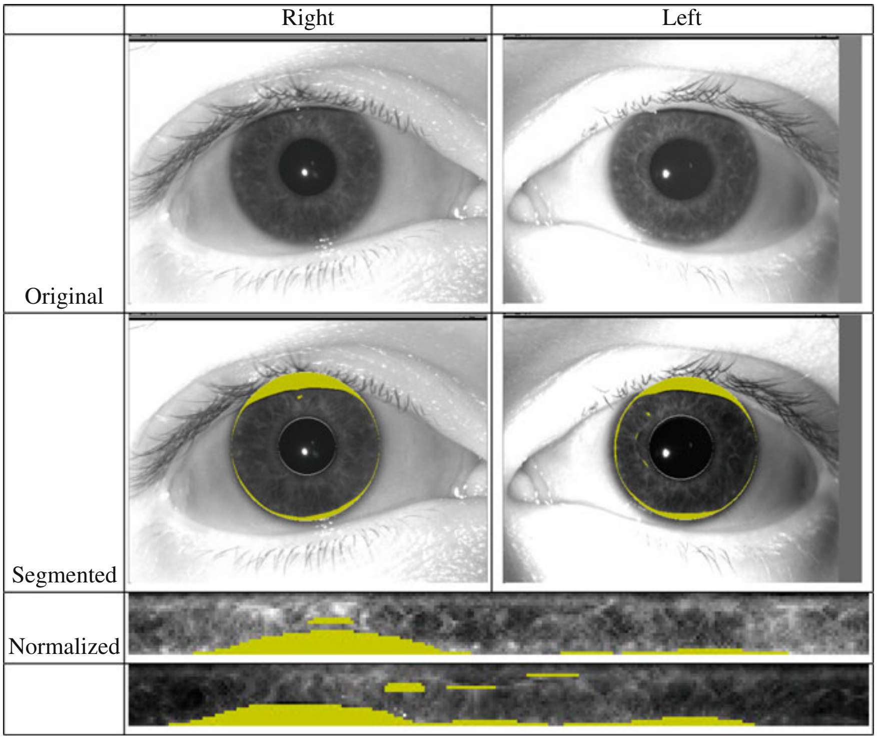


Fig. 9.7 Example of iris images captured using an LG 4000 sensor

The second dataset has one left eye image and one right eye image for each of 750 males and 750 females, for a total of 3,000 images. Of the 1,500 distinct persons in the dataset, visual inspection of the images indicates that about one-fourth are wearing clear contact lenses. In order to improve the results we used the fusion of the left and right normalized iris images, totalling 3,000 normalized images with their corresponding segmentation mask computed using Osiris implementation.[1] This dataset is available to other researchers at GFI gender-from-iris dataset.[2]

A training portion of this 3,000-person dataset was created by randomly selecting 60% of the males and 60% of the females. The remaining 40% was used as the test set. Once parameter selection is finalized, the same validation set used in [27] was used for the final evaluation. This validation dataset contains 1,944 images: three left eye images and three right eye images for each of 175 males and 149 females. It is known that some subjects are wearing clear contact lenses, and evidence of this is visible in some images. Also, a few subjects are wearing cosmetic contact lenses in

[1]Biosecure project. http://biosecure.it-sudparis.eu/AB/.

[2]https://sites.google.com/a/nd.edu/public-cvrl/data-sets.

some images. In order to replicate real life conditions we did not remove any images containing any type of defect (blurred or noised).

It is important to note that iris images of different persons, or even the left and right iris images for a given person, may not present exactly the same imaging conditions. The illumination by the LEDs may come from either side of the sensor, specular highlights may be present in different places in the image, the inclination of the head may be different, the eyelid occlusion may be different, and so forth. Also the training, test and validation subsets are all independent and person disjoint in this work.

9.5 Experiments

Our method is implemented using the Theano[3] and Keras[4] open-source framework. Training was performed on a EC2 Amazon GPU machine with GRID K-520 using g2.2x large instances with 8GB of video memory. For DBN we needed to train two different stages, the pretrained stage with RBM and fine tuning with Deep MLP. The first took 207 min to train and the second 150 min.

Training each convolutional network (CNN-1 and CNN-2) required about four hours with 238 s per epoch to CNN-1 and 10 h for CNN-2. Predicting gender on a single normalized image using our network requires about 100 ms. for CNN-1 and 178 ms. for CNN-2. Training running times can conceivably be substantially improved by running the training on image batches.

9.6 Hyperparameters Selection

In this work, we used grid search to find the best hyperparameters of the DBN + Deep MLP and Lenet-5 CNN. We look at tuning the batch size and number of epochs used when fitting the networks. The batch size in iterative gradient descent is the number of patterns shown to the network before the weights are updated. It is also an optimization in the training of the network, defining how many patterns to read at a time and keep in memory.

The number of epochs is the number of times that the entire training dataset is shown to the network during training. Some networks are sensitive to the batch size, such as Multilayer Perceptron Networks, and Convolutional Neural Networks.

In this work, we look at tuning the batch size and number of epochs used when fitting the network. We evaluated a suite of different mini batch sizes from $n = 8$ to $n = 500$ in steps of 2^n for CNN-1–2 and $m = 10$ to $m = 100$ in steps of 10.

[3]http://deeplearning.net/software/theano/NEWS.html.

[4]https://keras.io/backend/.

Keras and Theano offers a suite of different state-of-the-art optimization algorithms such as:'SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', and others. We tuned the optimization algorithm used to train the network, each with default parameters. The best result was reached with 'SDG'.

The Learning rate parameter controls how much to update the weight at the end of each batch and the momentum controls how much to let the previous update influence the current weight update. We tried an small set of standard learning rates and momentum values ranging from 0.1 to 0.9 in steps of 0.1 for CNN-1 and CNN-2. For DBN a learning rate parameter for the pretraining stage and another learning rate parameter for the fine tuning stage are needed. Both were set in the range $1e-1$ to $1e-5$.

The activation function controls the non-linearity of individual neurons and when to fire. Generally, the rectifier activation function is the most popular, but $sigmoid$ and $tanh$ functions were also very used and these functions may still be more suitable for different problems. In this work, we evaluated the suite of different activation functions available. We only used these functions in the hidden layer, as we required a sigmoid activation function in the output for the binary classification problem to reach the best results in Deep MLP. For CNNs the best results were reach with $ReLU$ and $Softmax$ activation functions.

We also looked at tuning the dropout rate for regularization parameter in an effort to limit overfitting and improve the models ability to generalize. To get good results, dropout is best combined with a weight constraint such as the max norm constraint. This involves fitting both the dropout percentage and the weight constraint. We tried dropout percentages between 0.0 and 0.9 and max norm weight constraint values between 0 and 5. The best results were reached with 0.5.

The number of neurons in a layer also is an important parameter to tune. Generally the number of neurons in a layer controls the representational capacity of the network, at least at that point in the topology. The best number of neurons is shown in Tables 9.2, 9.3.

Table 9.2 Gender classification rates using DBN + Deep MLP. The number of the hidden layers are indicated in parenthesis. We best results were reached with batch size of 100. 'preT-LR' represent pre-trainning Learning Rate stage with value of $1 * e - 2$ and 'FineT' represent Fine Tunning Learning rate with value of $1 * e - 1$

Number and size of hidden layers	Accuracy (%)
(2) 5–5K	50.67
(2) 5–10K	55.67
(2) 10–5K	59.67
(2) 10–10K	61.00
(3) 5–5–5K	62.67
(3) 10–10–10K	66.79
(5)10–5–10–5–10K	**77.79**

Table 9.3 Gender classification rates using CNN-1 with Batch size of 64. LR represent Learning Rate

LR	Epochs	Accuracy (%)
1*e-1	50	50.08
1*e-2	50	49.92
1*e-3	50	62.77
1*e-4	50	62.94
1*e-5	50	64.94
1*e-1	100	51.10
1*e-2	100	50.42
1*e-3	100	63.94
1*e-4	100	69.11
1*e-5	100	69.27
1*e-1	200	53.08
1*e-2	200	56.08
1*e-3	200	63.27
1*e-4	200	67.77
1*e-5	200	70.62
1*e-1	500	53.08
1*e-2	500	57.00
1*e-3	500	65.27
1*e-4	500	69.22
1*e-5	500	**77.91**
1*e-1	750	63.43
1*e-2	750	65.25
1*e-3	750	68.28
1*e-4	750	67.37
1*e-5	750	77.00
1*e-1	1000	54.14
1*e-2	1000	66.87
1*e-3	1000	71.11
1*e-4	1000	64.55
1*e-5	1000	68.98

9.7 Results

9.7.1 Semi-supervised Method

The RBM pretraining stage is very sensitive to batch size and learning rate, therefore we used values from 1 to 200 batch size and learning rate from 1e-1 to 1e-5 for

Table 9.4 Gender classification rates using CNN-2 with Batch size of 64. LR represent Learning Rate

LR	Epochs	Accuracy (%)
1*e-1	50	61.03
1*e-2	50	61.33
1*e-3	50	63.33
1*e-4	50	65.33
1*e-5	50	64.26
1*e-1	100	62.33
1*e-1	100	63.03
1*e-3	100	65.33
1*e-4	100	66.33
1*e-5	100	68.00
1*e-1	200	65.33
1*e-2	200	66.00
1*e-3	200	66.33
1*e-4	200	70.00
1*e-5	200	73.00
1*e-1	500	65.33
1*e-2	500	66.08
1*e-3	500	79.00
1*e-4	500	81.00
1*e-5	**500**	**83.00**
1*e-1	750	67.33
1*e-2	750	67.00
1*e-3	750	79.66
1*e-4	750	80.33
1*e-5	750	81.00

pre-train with 10 K images and fine tuning stage with near to 4,900 images. In average the pretrained stage takes 307 min and fine tuning only 150 min. Table 9.2 shows a summary of the results.

9.7.2 Supervised Method

In order to compare the results with the semi-supervised method we trained two ‘small’ Lenet-5 CNN. We consider these networks ‘small’, because we do not have a huge quantity of iris images available like in databases such as Deepface, Imagenet, or VGG(16–19). This is one of our main concerns for applying deep-learning techniques to any iris-based soft biometric problem. A future challenge to researchers in this area

Table 9.5 Accuracy of our proposal models (*) and the state-of-the-art methods

Paper	Accuracy (%)	Obs.
Thomas et al. [28]	80.00	Discard poor quality
Lagree et al. [17]	62.00	
Bansal et al. [2]	83.06	Only 300 images
Tapia et al. [26]	91.00	Non person disjoint
Costa-Abreu et al. [8]	89.74	
Bobeldyk et al. [4]	65	Normalized images
Tapia et al. [27]	89.00	Binary images
CNN-1(*)	77,91	Supervised
CNN-2(*)	**83.00**	Supervised
DBM+Deep MLP (*)	77.79	Semi-supervised

Table 9.6 Summary of the best gender classification rates. Second column shows results without data augmentation. Third column shows results with data augmentation (DA) in relation to Tables 9.3 and 9.4

DNN	Accuracy (%)	Accuracy – DA (%)
(5)10–5–10–5–10K	77.79	78.79
CNN-1	77.91	79.03
CNN-2	83.00	84.66

is to increase the number of soft biometric images captured with different sensors to train more complex models.

Tables 9.2, 9.3 and 9.4 show the summary of the results using different number of parameters. Table 9.5 shows the results of state-of-the-art algorithms compared with our two proposals.

In order to create a larger dataset to train a supervised stage, we used data augmentation by flipping vertically and horizontally the original iris images, increasing the number of images three times for each iris. We applied this technique only to the best results of Tables 9.3 and 9.4. As future work we propose to study other techniques to analyze the influence of data augmentation on the results. The summary of the results using data augmentation, are presented on Table 9.6.

9.8 Conclusion

Our best approach was to use convolutional neural networks trained with labeled iris data, that reached 83.00% without data augmentation and 84.66% with data augmentation being one of the top performance on relation with other state-of-the-art methods as shown in Table 9.5. We did not outperform our previous work in [27] but our deep-learning method presented here, did not use any preprocessing stages or feature selection methods. Also, this proposal used normalized images and not the encoded information. According with the results reported by Bobeldik et al. [4] normalization process may be filtering out useful information. We think these are good results considering the quantity of images used to train our methods. Deep-learning techniques usually require much larger databases to reach its full potential. We think that this work further validates the hypothesis about the iris having information about a person's gender.

Also, it is important to highlight the results reached by the semi-supervised method even when it is not reached the best results of the test. The accuracy is very motivating considering that the results were reached with unlabeled iris images.

One of the main contributions of this work is that, to our understanding, this is the first work that uses deep-learning techniques (both semi-supervised and supervised) to try to solve the problem of gender-from-iris. Our proposal reached good performance by training a common deep-learning architecture designed to avoid overfitting due to the limitation of available labeled iris images. Our network is comparable to some of the recent network architectures, thereby reducing the number of its parameters and the chance for overfitting.

Also, we increased the number of images in our training set using data augmentation in order to preliminary test its usefulness in improving the gender recognition process. The resulting techniques were tested using unfiltered iris images and perform comparably to most of the recent state-of-the-art methods.

Another contribution was to propose the usage of the large quantity of unlabeled iris images available to help in the initialization of the weights of a deep neural network. A larger number of unlabeled image may help to improve the gender classification rate. If we have more images captured from different sensors we can create a general purpose gender classification method and understand if the information present on the iris is dependent of the sensor used. To our understanding the features present on the iris that make gender classification feasible, are independent of the sensor used to capture the image. As a future challenge for all iris researchers, we propose to make an effort to capture more images with soft biometrics labeled data.

Two important conclusions can be made from our results. First, convolutional neural networks can be used to provide competitive iris gender classification results, even considering the much smaller size of contemporary unconstrained image sets labeled for gender. Second, the simplicity of our model implies that more elaborate systems using more training data may be capable of substantially improving results beyond those reported here.

Acknowledgements Thanks to Vince Thomas, Mike Batanian, Steve Lagree, Yingjie, Bansal and Costa Abreu for the work they have previously done in this research topic and Also to Professor Kevin Bowyer at University of Notre Dame for providing the databases. This work has been supported by Universidad Andres Bello, Faculty of Engineering, Department of Engineering Sciences (DCI).

References

1. L.A. Alexandre, Gender recognition: a multiscale decision fusion approach. Pattern Recognit. Lett. **31**(11), 1422–1427 (2010)
2. A. Bansal, R. Agarwal, R.K. Sharma, SVM based gender classification using iris images, in *Fourth International Conference on Computational Intelligence and Communication Networks* (CICN, 2012), pp. 425–429
3. Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks (2007), pp. 153–160
4. D. Bobeldyk, A. Ross, Iris or periocular? Exploring sex prediction from near infrared ocular images, in *Lectures Notes in Informatics (LNI), Gesellschaft fur Informatik, Bonn* (2016)
5. K.W. Bowyer, K. Hollingsworth, P.J. Flynn, Image understanding for iris biometrics: a survey. Comput. Vis. Image Underst. **110**(2), 281–307 (2008)
6. CANPASS, Canadian border services agency, CANPASS (1996)
7. M.A. Carreira-Perpinan, G.E. Hinton, On contrastive divergence learning (2005)
8. M.D. Costa-Abreu, M. Fairhurst, M. Erbilek, Exploring gender prediction from iris biometrics. Int. Conf. Biom. Spec. Interest Group (BIOSIG) **2015**, 1–11 (2015)
9. J. Daugman, How iris recognition works. IEEE Trans. Circuits Syst. Video Technol. **14**(1), 21–30 (2004)
10. J. Daugman, Iris recognition at airports and border-crossings, in *Encyclopedia of Biometrics* (2009), pp. 819–825
11. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, ImageNet: a large-scale hierarchical image database, in *CVPR09* (2009)
12. K. Fukushima, Neocognitron: a self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biol. Cybern. **36**, 193–202 (1980)
13. G.E. Hinton, R.R. Salakhutdinov, Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006). doi:10.1126/science.1127647, http://www.ncbi.nlm.nih.gov/sites/entrez?db=pubmed&uid=16873662&cmd=showdetailview&indexed=google
14. M. Hubel, T.N. Wiesel, *Brain and Visual Perception* (Oxford Univeristy Press, Oxford, 2005)
15. F. Juefei-Xu, E. Verma, P. Goel, A. Cherodian, M. Savvides, Deepgender: occlusion and low resolution robust facial gender classification via progressively trained convolutional neural networks with attention, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops* (2016)
16. J. Kannala, E. Rahtu, BSIF: binarized statistical image features, in *ICPR* (IEEE Computer Society, 2012), pp. 1363–1366
17. S. Lagree, K. Bowyer, Predicting ethnicity and gender from iris texture, in *IEEE International Conference on Technologies for Homeland Security* (2011), pp. 440–445
18. Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, L.D. Jackel, Backpropagation applied to handwritten zip code recognition. Neural Comput. **1**(4), 541–551 (1989). doi:10.1162/neco.1989.1.4.541
19. G. Levi, T. Hassncer, Age and gender classification using convolutional neural networks, in *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (CVPRW, 2015), pp. 34–42. doi:10.1109/CVPRW.2015.7301352
20. E. Makinen, R. Raisamo, Evaluation of gender classification methods with automatically detected and aligned faces. IEEE Trans. Pattern Anal. Mach. Intell. **30**(3), 541–547 (2008a)
21. H. Peng, F. Long, C. Ding, Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. IEEE Trans. Pattern Anal. Mach. Intell. **27**(8), 1226–1238 (2005)
22. C. Perez, J. Tapia, P. Estevez, C. Held, Gender classification from face images using mutual information and feature fusion. Int. J. Optomech. **6**(1), 92–119 (2012)
23. P.J. Phillips, W.T. Scruggs, A.J. OToole, P.J. Flynn, K.W. Bowyer, C.L. Schott, M. Sharpe, FRVT 2006 and ICE 2006 large-scale results. IEEE Trans. Pattern Anal. Mach. Intell. **32**, 831–846 (2010)

24. J. Tapia, C. Perez, Gender classification based on fusion of different spatial scale features selected by mutual information from histogram of LBP, Intensity, and Shape. IEEE Trans. Inf. Forensics Secur. **8**(3), 488–499 (2013)
25. J.E. Tapia, C.A. Perez, Gender classification using one half face and feature selection based on mutual information, in *2013 IEEE International Conference on Systems, Man, and Cybernetics* (2013), pp. 3282–3287. doi:10.1109/SMC.2013.559
26. J.E. Tapia, C.A. Perez, K.W. Bowyer, Gender classification from iris images using fusion of uniform local binary patterns, in *European Conference on Computer Vision-ECCV, Soft Biometrics Workshop* (2014)
27. J. Tapia, C. Perez, K. Bowyer, Gender classification from the same iris code used for recognition. IEEE Trans. Inf. Forensics Secur. **11**(8), 1 (2016)
28. V. Thomas, N. Chawla, K. Bowyer, P. Flynn, Learning to predict gender from iris images. First IEEE Int. Conf. Biom.: Theory Appl. Syst. BTAS **2007**, 1–5 (2007)
29. UIDAI, Unique identification authority of India (2014)
30. P. Vincent, H. Larochelle, Y. Bengio, P.A. Manzagol, Extracting and composing robust features with denoising autoencoders, in *Proceedings of the 25th International Conference on Machine Learning, ICML '08* (ACM, New York, 2008), pp. 1096–1103. doi:10.1145/1390156.1390294
31. M.H. Yang, B. Moghaddam, Gender classification using support vector machines. Proc. Int. Image Process. Conf. **2**, 471–474 (2000)

Chapter 10
Deep Learning for Tattoo Recognition

Xing Di and Vishal M. Patel

Abstract Soft biometrics are physiological and behavioral characteristics that provide some identifying information about an individual. Color of eye, gender, ethnicity, skin color, height, weight, hair color, scar, birthmarks, and tattoos are examples of soft biometrics. Several techniques have been proposed to identify or verify an individual based on soft biometrics in the literature. In particular, person identification and retrieval systems based on tattoos have gained a lot of interest in recent years. Tattoos, in some extent, indicate ones personal beliefs and characteristics. Hence, the analysis of tattoos can lead to a better understanding of ones background and membership to gang and hate groups. They have been used to assist law enforcement in investigations leading to the identification of criminals. In this chapter, we will provide an overview of recent advances in tattoo recognition and detection based on deep learning. In particular, we will present deep convolutional neural network-based methods for automatic matching of tattoo images based on the AlexNet and Siamese networks. Furthermore, we will show that rather than using a simple contrastive loss function, triplet loss function can significantly improve the performance of a tattoo matching system. Various experimental results on a recently introduced Tatt-C dataset will be presented.

10.1 Introduction

Soft biometrics are physiological and behavioral characteristics that provide some identifying information about an individual [6]. Color of eye, gender, ethnicity, skin color, height, weight, hair color, scar, birthmarks, and tattoos are examples of soft biometrics. Several techniques have been proposed to identify or verify an individ-

X. Di · V.M. Patel (✉)
Rutgers University, 94, Brett Road, Piscataway, NJ 08854, USA
e-mail: vishal.m.patel@rutgers.edu

X. Di
e-mail: xing.di@rutgers.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_10

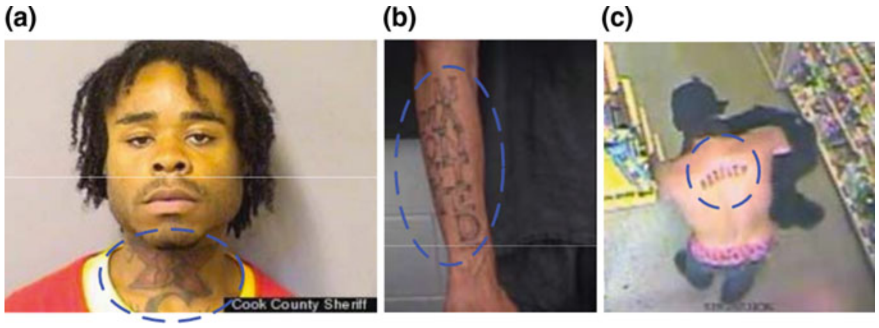


Fig. 10.1 Practical use of tattoos in arrest of suspects [11]

ual based on soft biometrics [6, 14, 29, 33] in the literature. In particular, person identification and retrieval systems based on tattoos have gained a lot of interest in recent years [12, 15, 23, 26]. Tattoos, in some extent, indicate one's personal beliefs and characteristics. Hence, the analysis of tattoos can lead to a better understanding of one's background and membership to gang and hate groups [23]. They have been used to assist law enforcement in investigations leading to the identification of criminals [27]. For instance, Fig. 10.1 shows three different cases where tattoos were used to identify and apprehend the suspects.

A number of different methods have been proposed in the literature for tattoo detection, recognition, clustering, and retrieval [1, 2, 8, 11–13, 15, 16, 18, 19, 22–25, 36]. Previous approaches essentially tackle these problems by first extracting some sort of generative or discriminative features from the given images and then training discriminative classifiers for matching. The performance of these methods is limited by the strength of the features they use. In previous approaches, the features used are often hand-crafted such as Gabor, LBP, or SIFT [23, 27]. In recent years, features obtained using deep convolutional neural networks (CNNs) have yielded impressive results on various computer vision applications such as object detection [10, 28, 31] and recognition [4, 20]. Recent studies have shown that in the absence of massive datasets, transfer learning can be effective as it allows one to introduce deep networks without having to train them from scratch [38]. For instance, one can use deep CNNs such as AlexNet [20] or Siamese network [3, 5] pretrained with a large generic dataset such as ImageNet [30] as meaningful feature extractors. In this chapter, we review a few recent approaches to tattoo detection and recognition based on deep CNN methods [9, 37].

This chapter is organized as follows. Section 10.2 gives an overview of the recently introduced NIST Tott-C tattoo dataset. Details of various CNN networks for tattoo detection and recognition are given in Sect. 10.3. The performance of different methods on the Tott-C dataset are compared in Sect. 10.4. Finally, Sect. 10.5 concludes the chapter with a brief summary and discussion.

10.2 NIST Tott-C Dataset

In order to promote research and development in tattoo-based recognition applications, a tattoo dataset called Tattoo Recognition Technology - Challenge (Tatt-C) was recently developed by NIST [26, 27]. This dataset contains a total of 16,716 tattoo images collected operationally by law enforcement and is partitioned into five use cases derived from operational scenarios. These use cases are as follows

- Tattoo Identification: matching different instances of the same tattoo image from the same subject over time,
- Region of Interest: matching a subregion of interest that is contained in a larger tattoo image,
- Mixed Media: matching visually similar or related tattoos using different types of images (i.e., sketches, scanned print, computer graphics, and graffiti),
- Tattoo Similarity: matching visually similar or related tattoos from different subjects,
- Tattoo Detection: detecting whether an image contains a tattoo or not.

In this chapter, we mainly focus on the following three use cases—tattoo detection, tattoo similarity, and mixed media. Figure 10.2 shows samples images from the Tatt-C dataset corresponding to these use cases. Tattoo detection has several implications in database maintenance and construction when a dataset consists of weakly

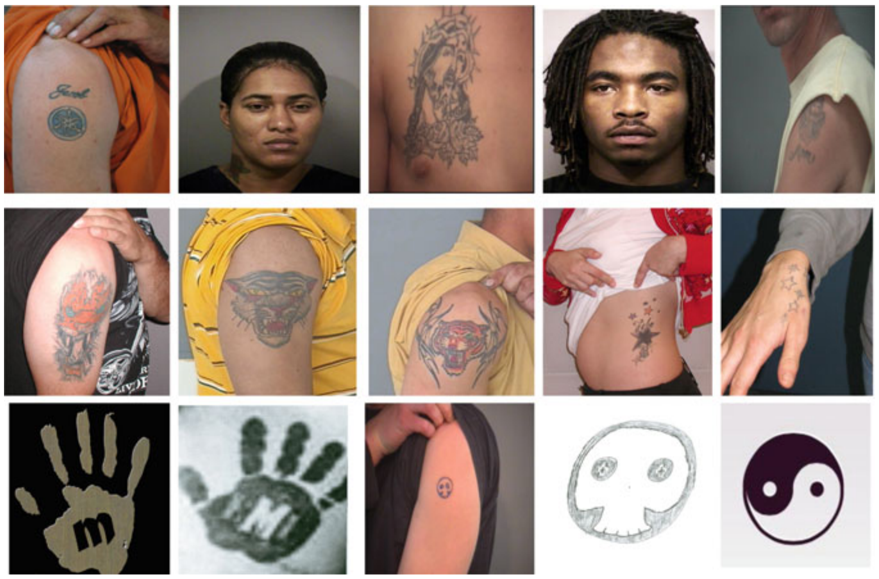


Fig. 10.2 Samples images from the Tatt-C database. *1st row* images corresponding to the tattoo detection use case. *2nd row* images corresponding to the tattoo similarity use case. *3rd row* images corresponding to the mixed media use case

labeled data. Partially or ambiguously labeled data often makes the automatic interpretation and extraction of different types of images challenging. As was indicated in [27], in the ANSI/NIST Type 10 record, facial mugshot images and scar, mark, tattoo images are stored in the same record type. If some data are mislabeled or unlabeled, then automatic extraction of the data based on image content becomes a major issue.

Tattoo similarity is another use case that has applications in group or gang affiliations. Since members of a group or gang tend to have similar tattoos, one can try to identify individuals belonging to the same gang by looking for people with similar tattoos. In this use case, the objective is to match a probe image with one or more gallery images.

Mixed media is the use case that has application in investigative intelligence gathering where the tattoo is not necessarily captured by a camera but described as a sketch. In this test case, data consists of mixed media and tattoo images and given a mixed media probe image, one has to match one or more tattoos in the dataset [27].

From the use cases described above, we can see that tattoo detection is a two class classification problem and the other two cases, mixed media and tattoo similarity, are both one-to-many verification problems. Hence, one can use deep CNNs to solve these problems. In particular, [7] studied the performance of deep CNN features on tattoo recognition problems. For the classification problems, such as tattoo detection, fine-tuned deep features were extracted based on the AlexNet network using the tattoo images from the Tatt-C dataset and a linear SVM was trained for classification. For the verification problems, deep features were extracted using the Siamese network and data were matched using the Euclidean distance as well as a measure based on a triplet loss function. In [37] a network similar to the AlexNet was used for tattoo detection.

10.3 CNNs for Tattoo Detection and Recognition

In this section, we describe the details of the method proposed in [7] for tattoo recognition based on the AlexNet [21] and the Siamese [5] networks.

10.3.1 Deep Tattoo Detection

The method proposed for tattoo detection framework in [7] consists of two main stages. In the first stage, they extracted the deep features based on the AlexNet framework. Figure 10.3 shows the AlexNet architecture. Then, in the second stage, they trained a linear SVM to determine whether a given image contains tattoo or not. The deep CNN model was implemented using caffe [17]. As the AlexNet has been trained on the ImageNet LSVRC-2010 database [30], they fine-tuned the network on the Tatt-C dataset for tattoo detection [26].

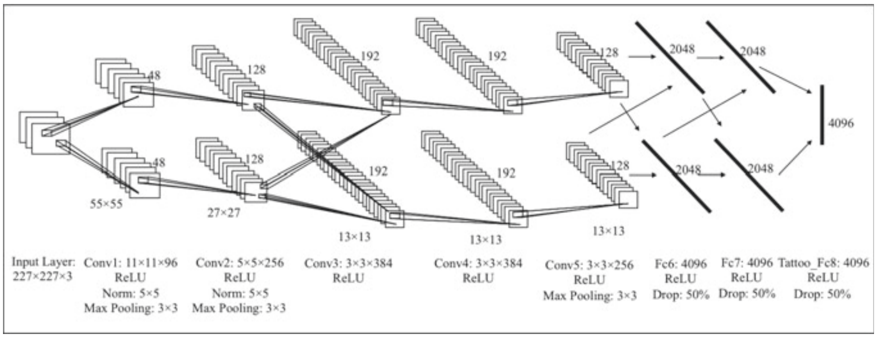


Fig. 10.3 The AlexNet architecture

Table 10.1 The AlexNet architecture used in [7]

Name	Type	Filter size/stride	Output size
Conv1	Convolution	$11 \times 11/4$	$55 \times 55 \times 96$
Relu1	ReLU		$55 \times 55 \times 96$
Norm1	LRN	5×5	$55 \times 55 \times 96$
Pool1	Max pooling	$3 \times 3/2$	$27 \times 27 \times 96$
Conv2	Convolution	$5 \times 5(pad2)/1$	$27 \times 27 \times 256$
Relu2	ReLU		$27 \times 27 \times 256$
Norm2	LRN	5×5	$27 \times 27 \times 256$
Pool2	Max pooling	$3 \times 3/2$	$13 \times 13 \times 256$
Conv3	Convolution	$3 \times 3(pad1)/1$	$13 \times 13 \times 384$
Relu3	ReLU		$13 \times 13 \times 38)$
Conv4	Convolution	$3 \times 3(pad1)/1$	$13 \times 13 \times 384$
Relu4	ReLU		$13 \times 13 \times 384$
Conv5	Convolution	$3 \times 3(pad1)/1$	$13 \times 13 \times 256$
Relu5	ReLU		$13 \times 13 \times 256$
Pool5	Max pooling	$3 \times 3/2$	$6 \times 6 \times 256$
Fc6	Fully connection		4096×1
Relu6	ReLU		4096×1
Drop6	Dropout	50%	4096×1
Fc7	Fully connection		4096×1
Relu7	ReLU		4096×1
Fc8_tattoo	Fully connection		2×1

Table 10.1 gives the details of the deep CNN architecture used for tattoo detection. All the images, during the training process are scaled into [0, 1] and subtracted from the their mean value. These training images are also flipped about the horizontal and vertical axis before feeding them into the network in order to increase the number of

training data for learning the network parameters. During the training and validation phases, they cropped the image into the standard 227 $\times$ 227 size. The basic learning rate for the tattoo detection was set equal to 10^{-4}. The decay rate, gamma, was selected to be 0.1 for every 3500 iterations. The multiplications of the convolutional layers are 1 for weights and 2 for biases. The weights values for the filter and were set randomly according to a Gaussian distribution with 0.01 standard deviation and weight for the bias is set equal to 1. The negative slope was set equal to 0 in the ReLU layer. The softmax loss layer computes the multinomial logistic loss of the softmax of its inputs. It is conceptually identical to a softmax layer followed by a multinomial logistic loss layer, but provides a more numerically stable gradient [17]. The momentum and total iteration numbers were set equal to 0.9 and 10500, respectively.

After fine-tuning the AlexNet on the tattoo database, the deep features were extracted as the output of the *fc*7 layer, which is a 4096 dimension vector. Then, a two-class linear SVM was implemented using vlfeat [34] to classify the probe images based on their deep features. The parameter lambda was set equal to 0.01, and the maximum number of iterations was set equal to 10^4.

The tattoo detection dataset has 2349 images, which includes the tattoo and non-tattoo images. Also, there is a ground_truth.txt file, which gives the labels "tattoo" or "non-tattoo" for each image. In this use case, they used label 1 to indicate the tattoo images, and label -1 to indicate the non-tattoo images. Following the standard protocol defined in [27], four out of five probe images were used for training and use the remaining images for testing. For instance, when testing on the 1st probe-list images, they used the images from the second, third, fourth, and fifth probe-lists for training. This process was repeated for all the probe-list images. Figure 10.4 shows the output from the first three convolutional layers corresponding to three sample

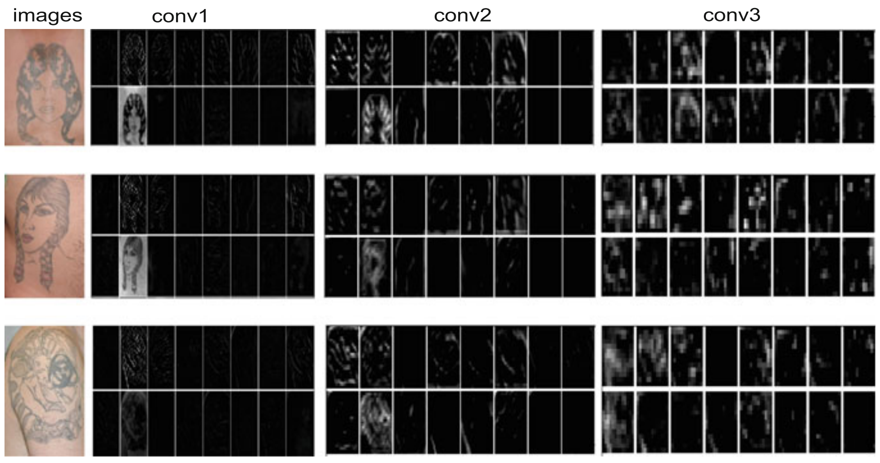


Fig. 10.4 Some feature maps from Conv1, Conv2, and Conv3 layers. The upper feature maps are more robust the illumination changes

images in the Tatt-C dataset. One can see that these features do capture meaningful information about tattoos such as edges, lines, and corner points.

Note that a deep CNN model similar to the AlexNet was recently proposed in [37] for tattoo detection on the NIST Tott-C dataset. The proposed model uses a private database containing 10,000 images collected from Flickr to train the network.

10.3.2 Deep Tattoo Recognition

For the tattoo verification cases such as tattoo similarity and tattoo mixed media use cases, the Siamese network was trained directly on the Tatt-C dataset. The Siamese network used in [7] is shown in Fig. 10.5 and details are given in Table 10.2. As before, the data augmentation was used by flipping the mixed media and tattoo similarity images horizontally and vertically and scaled them into [0, 1].

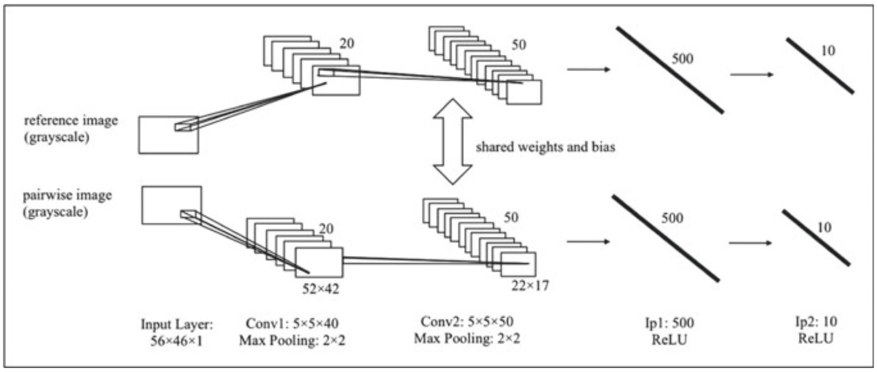

reference image (grayscale)
pairwise image (grayscale)
20
50
500
10
shared weights and bias
20
50
500
10
52×42
22×17
Input Layer: 56×46×1
Conv1: 5×5×40 Max Pooling: 2×2
Conv2: 5×5×50 Max Pooling: 2×2
Ip1: 500 ReLU
Ip2: 10 ReLU

Fig. 10.5 The Siamese network architecture

Table 10.2 Details of the Siamese network used in [7] for tattoo recognition

Name	Type	Filter size/stride	Output size
Conv1	Convolution	$5 \times 5/1$	$52 \times 42 \times 20$
Pool1	Pooling	$2 \times 2/2$	$26 \times 21 \times 20$
Conv2	Convolution	$5 \times 5/1$	$22 \times 17 \times 50$
Pool2	Pooling	$2 \times 2/2$	$11 \times 9 \times 50$
ip1	InnerProduct		500×1
relu1	ReLU		
ip2	InnerProduct		10×1
feat	InnerProduct		2×1

For the mixed media use case, the contrastive loss function [5] was used which is defined as

$$\begin{aligned} L(W) &= \sum_{i=1}^{P} L(W, (Y, X_1, X_2)^i), \\ L(W, (Y, X_1, X_2)^i) &= (1 - Y) L_G (E_W (X_1, X_2)^i) \\ &\quad + Y L_I (E_W (X_1, X_2)^i), \end{aligned} \tag{10.1}$$

where $(Y, X_1, X_2)^i)$ is the i-th sample which is composed of a pair of images (X_1, X_2) and a label Y, L_G is the partial loss function for a genuine pair, L_I the partial loss function for an impostor pair, and P is the number of training samples. In caffe, they used the Euclidean distance for $E_W(X_1, X_2)$. The margin they set in the training was 1. The total training iteration was set equal to 7×10^4. The initial learning rate was set equal to 10^{-4} and it decreases by 10% every 2×10^4 iterations. The multiplication learning rate for the neuron was set equal to 1 and 2 for the bias.

There are a total of 453 images (181 probe and 272 gallery) in the mixed media dataset. The "genuine pairwise" were made, which consisted of the probe images and their verified gallery images, and the "impostor pairwise", which consisted of the probe images and their unverified images. The number of "impostor pairwise" images were much larger than the "genuine pairwise" images. As a result, they randomly chose the equal number of "impostor pairwise" images and "genuine pairwise" images as the training subset. The images were cropped to 56×46. After training the network, output from the "ip2" layer was used as features. Finally, the images were verified based on the Euclidean distances.

For the tattoo similarity use case, rather than using the contrastive loss function, they replace it with the triplet loss function [32, 35] as it seemed to produce better results. The triplet loss function is defined as

$$\begin{aligned} L = \sum_{i=1}^{N} \max(0, &||f(x_i^a) - f(x_i^p)||_2^2 \\ &- ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha), \end{aligned} \tag{10.2}$$

where x_i^a is the reference image, x_i^p is the "genuine pairwise" image (positive pairwise), and x_i^n is the "impostor pairwise" (negative pairwise). The threshold α is referred to as "margin". In tattoo similarity case, we replace the contrastive loss function with the triplet loss function. We set the margin equal to 0.005 and the total iteration number to 4×10^4. All the parameters were the same as the original Siamese Network Configuration shown in Table 10.2 except that the dimension of "ip2" is 256 instead of 10. The initial learning rate was set equal to 0.0002 and decreases to 10% every 1×10^4 iterations. As before, the multiplication learning rates for the neuron is set equal to 1 and 2 for the bias. Tattoo similarity dataset has 2212 images, which consists of 851 probe images and 1361 gallery images. All the images for mixed media and tattoo similarity are gray-scaled before training. Output from the "ip2" layer is used as features.

10.4 Experimental Results

In this section, we present results of the deep CNN-based methods proposed in [7] for tattoo recognition on the Tatt-C dataset. The performance of the CNN-based methods were compared with those reported in [27]. The performance of different methods are compared using accuracy and Cumulative Match Characteristic (CMC) curves. Tattoo accuracy is defined as the number of correctly classified tattoo images TT, divided by the total number of tattoo images N_{tattoo} as

$$Tattoo\ accuracy = \frac{TT}{N_{tattoo}}. \tag{10.3}$$

Non-tattoo accuracy is defined as the number of correctly classified non-tattoo images NT, divided by the total number of non-tattoo images $N_{non-tattoo}$ as

$$Non\text{-}Tattoo\ accuracy = \frac{NT}{N_{non\text{-}tattoo}}. \tag{10.4}$$

The overall accuracy is defined as the sum of correctly classified tattoo and non-tattoo images divided by the total number of images

$$Overall\ accuracy = \frac{TT + NT}{N_{tattoo} + N_{non\text{-}tattoo}}. \tag{10.5}$$

The CMC is defined as the fraction of searches that return the relevant images as a function of the candidate list length. The longer the candidate list, the greater the probability that relevant images are on the list. For searches that have multiple relevant matches in the gallery, the cumulative accuracy or hit rate at any particular rank is calculated with the best-ranked match and represents a best-case scenario.

10.4.1 Tattoo Detection

The first row of Fig. 10.2 shows some sample images from the Tatt-C dataset corresponding to the detection use case. There are in total 2349 images in this subset—1349 tattoo images and 1000 non-tattoo images. The non-tattoo images are essentially face images extracted from the Multiple Encounter Database 2 (MEDS- II) [27]. The performance of different methods on the tattoo detection experiment is shown in Table 10.3. As can be seen from this table, the CNN-based method proposed in [7] outperforms the previous best non CNN-based methods reported in [27] and achieves the overall accuracy of 99.83%. It is interesting to note that even though both [7] and [37] use similar AlexNet type of network for tattoo detection, the performance of [7] is slightly better than that of [37]. This may be due to the fact that [37] used 10,000 images from Flickr and the variations that are present in that dataset are somewhat

Table 10.3 Performance comparison of different methods on the detection use case

Algorithm	Non-tattoo accuracy	Tattoo accuracy	Overall accuracy
CEA_1	0.988	0.932	0.956
Compass	0.386	0.798	0.622
MITRE_1	0.750	0.734	0.741
MITRE_2	0.948	0.924	0.934
MorphoTrak	0.950	0.972	0.963
Deep Tattoo [7]	0.9980	0.9985	0.9983
CNN [37]	0.989	0.987	0.988

different than the ones present in the Tott-C dataset. In contrast, apart from Tott-C and imagenet(pre-trained network), no external dataset was used in [7] to train the network.

In Fig. 10.6, we display the images on which our deep CNN algorithm fails to correctly detect a tattoo image. In particular, only 4 out of 2349 images were misclassified by the CNN-based algorithm. Two tattoo images were classified as non-tattoo images and two non-tattoo images were classified as tattoo images. In the first row

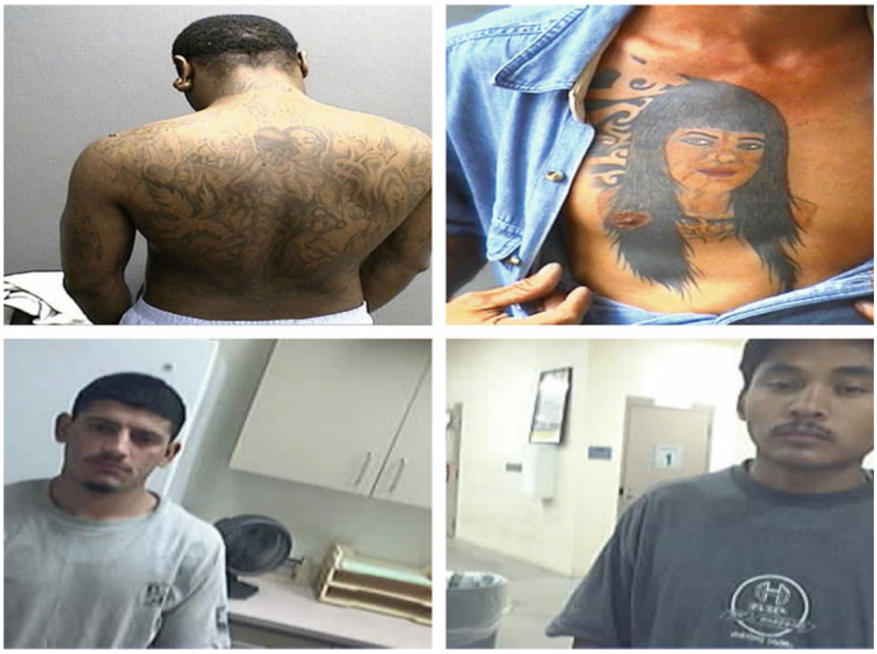


Fig. 10.6 Wrongly classified images. Only 4 out of 2349 images are wrongly classified by the deep CNN-based method [7]

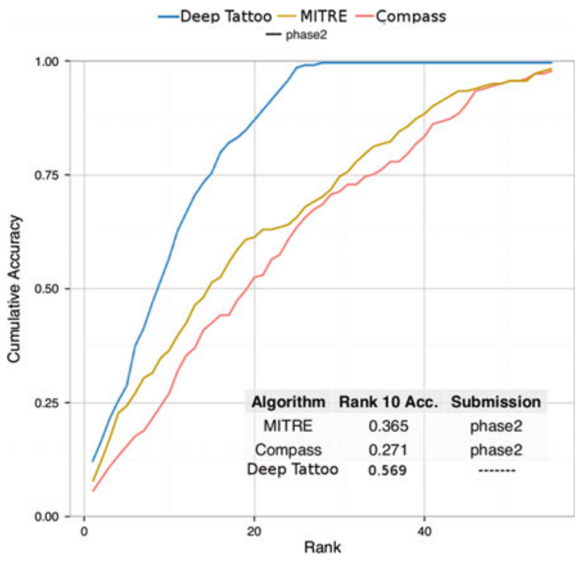
of this figure, a tattoo image is recognized as a face image and in the second row, two face images are recognized as tattoo images. As can be seen from this figure, the reason why the CNN-based method in [7] fails is that the wrongly classified tattoo image is a "face-like" image and the algorithm classifies it as a face image. Also, this happens to the face images, which is wrongly identified as tattoo image by the algorithm classifier.

10.4.2 *Mixed Media*

Results of different methods corresponding to the mixed media use case are shown in Table 10.4 and in Fig. 10.7. As can be seen from this table, the CNN-based method significantly outperforms the previous methods and achieves 100% accuracy at rank 28. The descriptor used by MITRE is the shape contexts-based and Compass uses

Table 10.4 Performance comparison of different methods on the mixed media use case. Number of probes: 181, Average gallery size: 55. The Submission column indicates where those accuracies come from as they were in tott-c [27]

Algorithm	Submission	Rank 1 accuracy	Rank 10 accuracy	Rank 20 accuracy	Rank 30 accuracy
Compass	Phase2	0.055	0.271	0.525	0.713
MITRE	Phase2	0.077	0.365	0.613	0.746
Deep Tattoo	- - - - -	0.122	0.569	0.873	1



Algorithm	Rank 10 Acc.	Submission
MITRE	0.365	phase2
Compass	0.271	phase2
Deep Tattoo	0.569	-------

Fig. 10.7 The CMC curves corresponding to different methods on the mixed media use case [7]

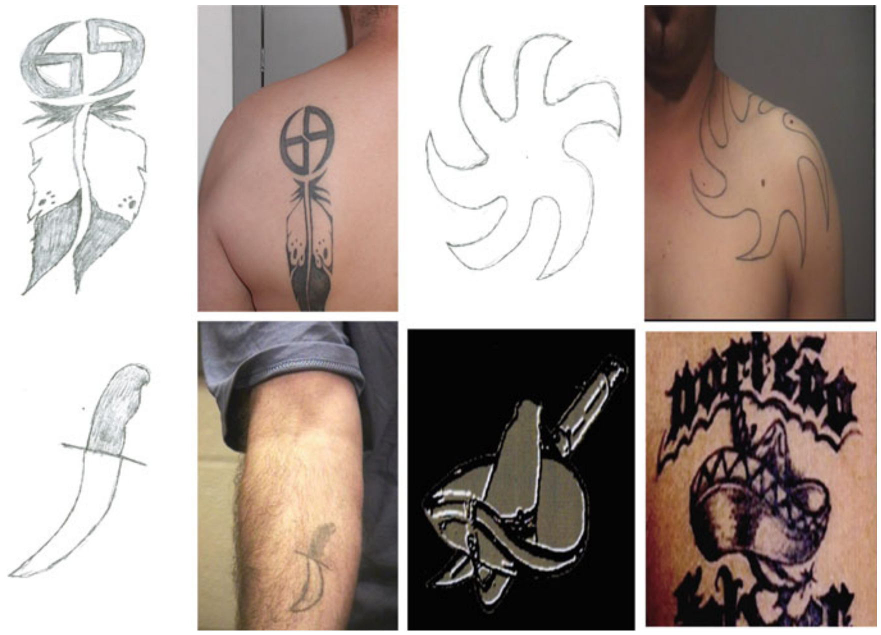


Fig. 10.8 Sample result from the mixed media use case [7]. *First row* correct matching. *Second row* failed matches

some low-level features like color, brightness, contrast, etc. In contrast, the method proposed in [7] uses deep features directly learned on the tattoo images. As a result, it is able to capture the salient information that is present in the tattoo images better than the other methods. This experiment clearly shows the significance of deep features compared to hand-crafted features for tattoo recognition.

To gain further insight into this method, in Fig. 10.8 we show some correctly matched and wrongly matched samples. First row displays images that are correctly classified and the second row displays images on which our method fails to correctly classify the mixed media images. Again the reason why the CNN-based method correctly classifies mixed media images as tattoo images is because they look very similar to the tattoo images. This can be clearly seen by comparing images shown in the second row of Fig. 10.8.

10.4.3 Tattoo Similarity

Table 10.5 and Fig. 10.9 show the results of different methods on the tattoo similarity use case. As can be seen from these results, the CNN-based method outperforms the previous methods especially when the triplet loss function incorporated within the framework. For instance, at rank-10, our method with triplet loss function gives

Table 10.5 Performance comparison of different methods on the tattoo similarity media use case [7]. Number of probes: 851, Average gallery size: 272. The Submission column indicates where those accuracies come from as they were in tott-c [27]

Algorithm	Submission	Rank 1 accuracy	Rank 10 accuracy	Rank 20 accuracy	Rank 30 accuracy
Compass	Phase2	0.005	0.074	0.147	0.199
MITRE	Phase2	0.035	0.149	0.239	0.309
Deep Tattoo	Triplet	0.055	0.164	0.249	0.316
Deep Tattoo	Non-triplet	0.017	0.111	0.155	0.210

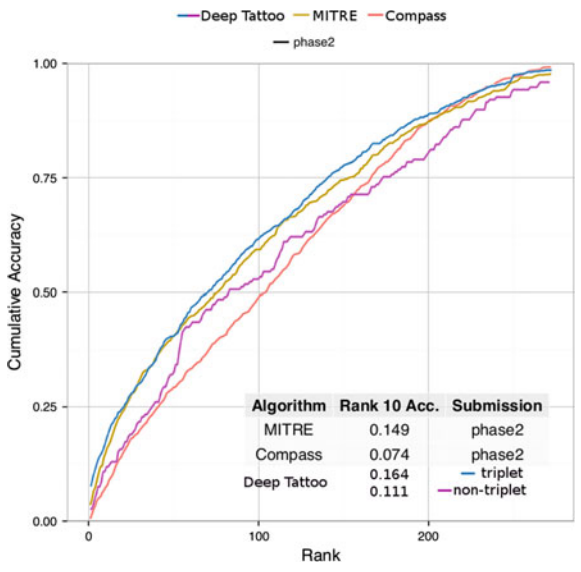


Fig. 10.9 The CMC curves corresponding to different methods on the tattoo similarity use case

an accuracy of 16.40% compared to 14.9, 7.4, and 11.1% for MITRE, Compass, and non-triplet based method. Again, this experiment clearly shows that one can significantly improve the performance of a tattoo recognition algorithm by using deep features.

In Fig. 10.10, we display a few correctly matched and incorrectly matched images for the tattoo similarity use case. First row of this figure shows the correctly matched images and the second row shows the incorrectly matched images. As can be seen from this figure, these images are extremely difficult to match as they contain various illumination, pose, and resolution variations. One of the reasons why the deep feature-based method does not work well in this particular use case is mainly due to the absence of a significant number of tattoo images with different variations to train the deep models.

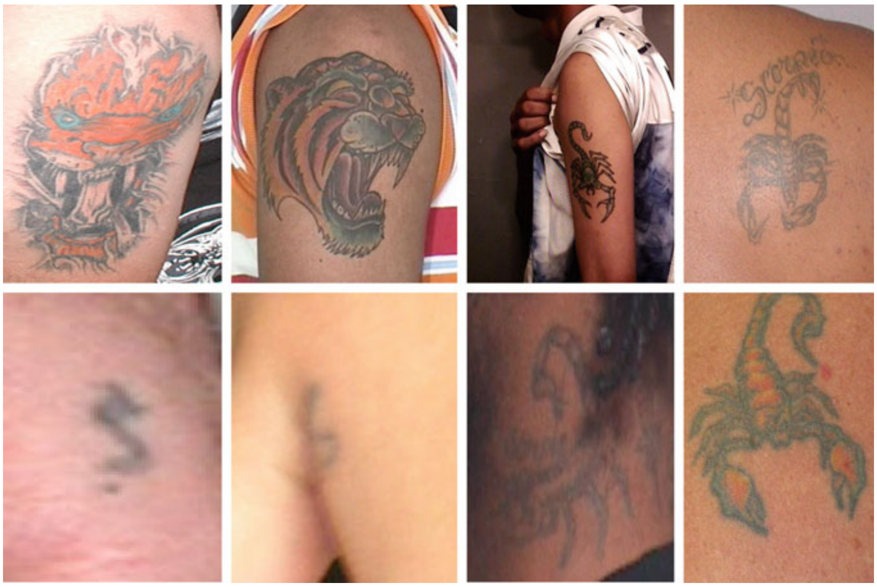


Fig. 10.10 Sample result from the mixed media use case. *First row* 629 correct matching. *Second row* failed matches

10.5 Conclusion and Future Directions

In this chapter, we presented deep feature-based methods for tattoo detection and recognition using the recently intruded AlexNet and Siamese networks. Furthermore, we showed that rather than using a simple contrastive loss function, triplet loss function can significantly improve the performance of a tattoo matching system based on deep features. Extensive experiments on the Tatt-C dataset demonstrated the effectiveness of different CNN-based approaches, like some other better network architectures rather than Siamese network.

Since deep learning is becoming increasingly important recently, one of the future directions for tattoo-based person recognition will be the classical methods combined with deep learning-based methods. The special statistical and geometrical properties of deep features will lead to new modeling techniques for tattoo recognition. Also, thanks to the fast developments in deep learning-based detection and landmark extraction techniques, tattoo detection, and alignment can be made more precise using these methods. As a result, they can provide geometrical model-based methods with improved performance. Another possible direction would be to build multitask CNN methods for doing joint detection, recognition, and retrieval. Deep multi-task leaning methods have gained a lot of interest in recent years and have been applied for various applications.

Acknowledgements This work was supported by an ARO grant W911NF-16-1-0126.

References

1. S. Acton, A. Rossi, Matching and retrieval of tattoo images: active contour cbir and glocal image features (2008), pp. 21–24
2. J.D. Allen, N. Zhao, J. Yuan, X. Liu, Unsupervised tattoo segmentation combining bottom-up and top-down cues. Proc. SPIE **8063**, 80,630L–80,630L–9 (2011)
3. J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, R. Shah, Signature verification using a "siamese" time delay neural network, in *Advances in Neural Information Processing Systems*, vol. 6, ed. by J.D. Cowan, G. Tesauro, J. Alspector (Morgan-Kaufmann, 1994)
4. J.C. Chen, V.M. Patel, R. Chellappa, Unconstrained face verification using deep cnn features, in *IEEE Winter Conference on Applications of Computer Vision* (2016)
5. S. Chopra, R. Hadsell, Y. LeCun, Learning a similarity metric discriminatively, with application to face verification, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR-2005*, vol. 1 (IEEE, 2005), pp. 539–546
6. A. Dantcheva, P. Elia, A. Ross, What else does your biometric data reveal? a survey on soft biometrics. IEEE Trans. Inf. Forensics Secur. **11**(3), 441–467 (2016)
7. X. Di, V.M. Patel, Deep tattoo recognition, in *IEEE Computer Society Workshop on Biometrics in Conjunction with CVPR-2016* (2016)
8. P. Duangphasuk, W. Kurutach, Tattoo skin detection and segmentation using image negative method, in *International Symposium on Communications and Information Technologies* (2013), pp. 354–359
9. P. Duangphasuk, W. Kurutach, Tattoo skin cross - correlation neural network, in *International Symposium on Communications and Information Technologies* (2014), pp. 489–493
10. R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in *Computer Vision and Pattern Recognition* (2014)
11. H. Han, A.K. Jain, Tattoo based identification: sketch to image matching, in *International Conference on Biometrics* (2013), pp. 1–8
12. B. Heflin, W. Scheirer, T. Boult, Detecting and classifying scars, marks, and tattoos found in the wild, in *International Conference on Biometrics: Theory, Applications and Systems* (2012), pp. 31–38
13. L. Huffman, J. McDonald, Mixed media tattoo image matching using transformed edge alignment, in *IEEE Symposium on Technologies for Homeland Security* (2016), pp. 1–6
14. A.K. Jain, U. Park, Facial marks: soft biometric for face recognition, in *IEEE International Conference on Image Processing* (2009), pp. 37–40
15. A. Jain, J.E. Lee, R. Jin, Tattoo-id: automatic tattoo image retrieval for suspect and victim identification, in *Advances in Multimedia Information Processing*, vol. 4810, Lecture Notes in Computer Science, ed. by H.S. Ip, O. Au, H. Leung, M.T. Sun, W.Y. Ma, S.M. Hu (Springer, Berlin, 2007), pp. 256–265
16. A.K. Jain, J.E. Lee, R. Jin, N. Gregg, Content-based image retrieval: an application to tattoo images, in *IEEE International Conference on Image Processing* (2009), pp. 2745–2748
17. Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: convolutional architecture for fast feature embedding (2014), arXiv:1408.5093
18. J. Kim, H. Li, J. Yue, E.J. Delp, Tattoo image retrieval for region of interest, in *IEEE Symposium on Technologies for Homeland Security* (2016), pp. 1–6
19. J. Kim, A. Parra, J. Yue, H. Li, E.J. Delp, Robust local and global shape context for tattoo image matching, in *IEEE International Conference on Image Processing* (2015), pp. 2194–2198
20. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks (2012), pp. 1097–1105
21. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Advances in Neural Information Processing Systems 25*, (Curran Associates, Inc. 2012), pp. 1097–1105, http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
22. J.E. Lee, A.K. Jain, R. Jin, Scars, marks and tattoos (SMT): soft biometric for suspect and victim identification, in *Proceedings of the Biometrics Symposium* (2008), pp. 1–8

23. J.E. Lee, R. Jin, A. Jain, W. Tong, Image retrieval in forensics: tattoo image database application. IEEE Multimed. **19**(1), 40–49 (2012)
24. J.E. Lee, R. Jin, A.K. Jain, Rank-based distance metric learning: an application to image retrieval, in *IEEE Conference on Computer Vision and Pattern Recognition* (2008), pp. 1–8
25. D. Manger, Large-scale tattoo image retrieval, in *2012 Ninth Conference on Computer and Robot Vision (CRV)* (2012), pp. 454–459
26. M. Ngan, P. Grother, Tattoo recognition technology - challenge (Tatt-C): an open tattoo database for developing tattoo recognition research, in *IEEE International Conference on Identity, Security and Behavior Analysis* (2015), pp. 1–6
27. M. Ngan, G.W. Quinn, P. Grother, Tattoo recognition technology–challenge (tatt-c) outcomes and recommendations. Technical report NISTIR 8078, National Institute of Standards and Technology (2015)
28. R. Ranjan, V.M. Patel, R. Chellappa, A deep pyramid deformable part model for face detection, in *IEEE International Conference on Biometrics Theory, Applications and Systems* (2015), pp. 1–8
29. D. Reid, S. Samangooei, C. Chen, M. Nixon, A. Ross, Soft biometrics for surveillance: an overview. Handb. Stat. **31**, 327–352 (2013)
30. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A.C. Berg, L. Fei-Fei, ImageNet large scale visual recognition challenge. Intern. J. Comput. Vis. (IJCV) **115**(3), 211–252 (2015). doi:10.1007/s11263-015-0816-y
31. S. Sarkar, V.M. Patel, R. Chellappa, Deep feature-based face detection on mobile devices, in *IEEE International Conference on Identity, Security and Behavior Analysis* (2016)
32. F. Schroff, D. Kalenichenko, J. Philbin, Facenet: a unified embedding for face recognition and clustering. CoRR (2015), arXiv:1503.03832
33. P. Tome, J. Fierrez, R. Vera-Rodriguez, M. Nixon, Soft biometrics and their application in person recognition at a distance. IEEE Trans. Inf. Forensics Secur. **9**(3), 464–475 (2014)
34. A. Vedaldi, B. Fulkerson, VLFeat: an open and portable library of computer vision algorithms (2008), http://www.vlfeat.org/
35. J. Wang, Y. Song, T. Leung, C. Rosenberg, J. Wang, J. Philbin, B. Chen, Y. Wu, Learning fine-grained image similarity with deep ranking, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2014), pp. 1386–1393
36. M.J. Wilber, E. Rudd, B. Heflin, Y.M. Lui, T.E. Boult, Exemplar codes for facial attributes and tattoo recognition, in *IEEE Winter Conference on Applications of Computer Vision* (2014), pp. 205–212
37. Q. Xu, S. Ghosh, X. Xu, Y. Huang, A.W.K. Kong, Tattoo detection based on cnn and remarks on the nist database, in *International Conference on Biometrics* (2016), pp. 1–7
38. J. Yosinski, J. Clune, Y. Bengio, H. Lipson, How transferable are features in deep neural networks? CoRR (2014), arXiv:1411.1792

Part IV
Deep Learning for Biometrics Security and Protection

Chapter 11
Learning Representations for Cryptographic Hash Based Face Template Protection

Rohit Kumar Pandey, Yingbo Zhou, Bhargava Urala Kota and Venu Govindaraju

Abstract In this chapter, we discuss the impact of recent advancements in deep learning in the field of biometric template protection. The representation learning ability of neural networks has enabled them to achieve state-of-the-art results in several fields, including face recognition. Consequently, biometric authentication using facial images has also benefited from this, with deep convolutional neural networks pushing the matching performance numbers to all time highs. This chapter studies the ability of neural networks to learn representations which could benefit template security in addition to matching accuracy. Cryptographic hashing is generally considered most secure form of protection for the biometric data, but comes at the high cost of requiring an exact match between the enrollment and verification templates. This requirement generally leads to a severe loss in matching performance (FAR and FRR) of the system. We focus on two relatively recent face template protection algorithms that study the suitability of representations learned by neural networks for cryptographic hash based template protection. Local region hashing tackles hash-based template security by attempting exact matches between features extracted from local regions of the face as opposed to the entire face. A comparison of the suitability of different feature extractors for the task is presented and it is found that a representation learned by an autoencoder is the most promising. Deep secure encoding tackles the problem in an alternative way by learning a robust mapping of face classes to secure codes which are then hashed and stored as the secure template. This approach overcomes several pitfalls of local region hashing and other face template algorithms. It also achieves state-of-the-art matching performance with a high standard of template security.

R.K. Pandey (✉) · Y. Zhou · B.U. Kota · V. Govindaraju
Univeristy of Buffalo, SUNY, Buffalo, NY 14260, USA
e-mail: rpandey@buffalo.edu

Y. Zhou
e-mail: yingbozh@buffalo.edu

B.U. Kota
e-mail: buralako@buffalo.edu

V. Govindaraju
e-mail: govind@buffalo.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*,
Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_11

11.1 Introduction

Privacy is a growing concern in today's world given the large digital footprint we leave behind on a day-to-day basis. This may be in the form of web browsing history, photos we share via social media, or passwords we register on applications and websites. Given the sensitive nature of most of this data, there are concerns of it falling into the wrong hands. Modern cryptography provides good solutions for the protection of sensitive information such as passwords and other textual data, but there are still forms of sensitive data that remain challenging to secure. This chapter focuses on the protection of one critical such data type; face biometric templates.

Biometric authentication is increasingly finding its way into our daily lives through face and fingerprint recognition systems in mobile phones as well as computers. Authentication based on "who we are" as opposed to "something we remember" or "something we possess" offers convenience and often, stronger system security. Typically, a biometric authentication system extracts and stores a template from the user's biometric data during the enrollment phase. During verification, the user's biometric is presented and a template is extracted and matched to the stored template. Depending on the matching score, access is granted or denied. One crucial, and often overlooked, aspect to such authentication is the protection of the stored template. If an attacker comes in possession of the stored template, not only can he gain access to the system, but also possibly recover the original biometric data of the user from it. Given that physical biometric features like face, fingerprint, or iris are not replaceable, compromising them would prevent the user from using them for any biometric authentication in the future.

11.2 Password Protection

Before getting into biometric template protection, let us briefly consider the most common form of authentication today; the string password. During enrollment, a template is extracted from the presented text and stored in the database. For text password authentication, the template is a one way non-invertible transform in the form of a hash digest generally computed using a cryptographic hash function like SHA. During verification, the password is presented again, its hash digest is calculated and compared to the stored hash digest. If the two passwords matched exactly, their hash digests would match as well, and access would be granted. If an attacker comes in possession of the template, the properties of hash functions like SHA make sure that no information is revealed about the original password. In fact, the attacker would have to brute force through each possible text password in order to find the one that corresponds to the template. Furthermore, it is straightforward to ask the user to change their password if the database breach is detected. Such hash-based template protection may seem ideal but comes at the high cost of requiring an exact match between the enrollment and verification texts.

11.3 Cryptographic Hash Functions

Hash functions are functions that map data of arbitrary size, generally referred to as the message, to a fixed length bit string called a hash digest. Cryptographic hash functions are a special case of hash functions with certain properties that make them suitable for cryptography. An ideal cryptographic hash function has the following properties:

1. It should be quick to compute the hash digest for any type of data.
2. It should be infeasible to obtain the original data given its hash digest.
3. It should be infeasible to find two messages with the same hash digest.
4. A small change in the message must lead to a large change in its hash digest in a manner such that the hash digest corresponding to the altered message is uncorrelated with the original digest.

Thus, the key aspect of cryptographic hash functions that differentiate them from other encryption methods is that the original data cannot be recovered from the encrypted form. Combined with the other properties mentioned above, the only possible attack on a hashed digest is a brute force attack, i.e., the attacker must go through all possible values of the message, hash each one, and compare the hash to the stored digest. In practice this task is made even harder by salting and multiple iterations of hashing. Adding a user-specific salt to the message prior to applying the hash function significantly increases the search space for the attacker. Whereas, applying several iterations of the hash function to the message increases the forward computation time of the digest and thus, significantly increases the time complexity for a brute force attack.

11.4 Biometric Template Protection

The goal of biometric template protection is to protect the biometric data in a manner similar to how text passwords are protected. Since the original biometric data is not changeable like a text password, its security is arguable more crucial. An ideal biometric template protection algorithm should possess the following properties.

1. Security—It should be infeasible to extract the original biometric data given the protected template.
2. Cancelability—It should be feasible to generate a new protected template from the original biometric if the old template has been compromised.
3. Performance—The template protection algorithm should not lead to loss in matching performance (FAR and FRR) of the biometric system.

From the perspective of security, cryptographic hash functions would be the ideal choice for biometric templates as well, but certain properties of biometric data make hash-based security very challenging. Hash-based security would require an exact

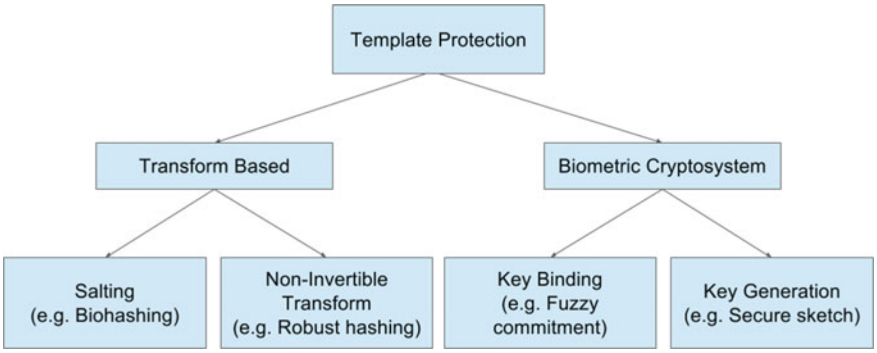

Template Protection
Transform Based
Biometric Cryptosystem
Salting (e.g. Biohashing)
Non-Invertible Transform (e.g. Robust hashing)
Key Binding (e.g. Fuzzy commitment)
Key Generation (e.g. Secure sketch)

Fig. 11.1 Biometric template protection algorithms

match between the templates extracted from different readings of biometric data. Since biometric data shows significant inherent variations between readings, it is very challenging to extract a templates that would match exactly. Even a single bit variation between the templates would lead to very different hash digest values. Additionally, since biometric data cannot be changed in the event of the hash digest being compromised, cancelability also needs to be addressed in some way.

Despite the difficulties involved in achieving hash-based template security for biometric data, several approaches have been proposed. As shown in Fig. 11.1, biometric template protection algorithms can be broadly classified into two types [12], biometric cryptosystems and feature transforms.

Biometric cryptosystem approaches rely on the use of some publicly available data (referred to as helper data) derived from the original biometric. The helper data should, of course, not reveal significant information about the original biometric data. Depending on the way the helper data is derived, biometric cryptosystems are of two types; key binding and key generation schemes. Key binding schemes derive the helper data by binding an external key with the biometric data. For example, fuzzy commitment binds a key in the form of an error correcting output code C, with the biometric data X, and generates helper data $X - C$ and $Hash(C)$. When a new sample X' needs to be verified, it is combined with the helper data yielding $C' = X' - (X - C)$. If X' is close to X, C' would be close to C and thus, the error correcting decoder will be able to correct C' to C. This enables us to match the hash of the corrected code to the stored $Hash(C)$. On the other hand, key generation schemes try to extract stable keys from the biometric data itself. Examples include fuzzy extractors and secure sketches proposed in [6]. The idea behind fuzzy extractors is to extract a uniform random string from the biometric data in a noise tolerant way. Thus even if the biometric data is altered slightly, the uniform random string can be reproduced exactly and used as a key. A secure sketch S, is some publicly visible data that is extracted from the biometric X, but does not reveal significant information about X. Given X' (some value close to X) and S it is possible to recover the value of X. Thus a secure sketch enables the exact recovery of the enrolled biometric but does not address uniformity like fuzzy extractors.

Feature transform based approaches simply try to transform the biometric data into an encoded space where matching is possible. They are of two types; non-invertible transforms and biometric salting. Non-invertible transforms apply a non-invertible function to the biometric data to generate a template from which it is infeasible to extract the original biometric. During verification the same function is applied to the new sample and matching is performed in the transformed space. Biometric salting based approaches transform the biometric data in an invertible manner using a secret key. Since it is possible to obtain the original data if the key is compromised, the security of salting-based approaches is entirely dependent on the security of the secret key.

Let us briefly go over some of these algorithms that have been applied to faces. Schemes that used cryptosystem based approaches include Fuzzy commitment schemes by Ao and Li [1], Lu et al. [16] and Van Der Veen et al. [30], and fuzzy vault by Wu and Qiu [32]. In general, the fuzzy commitment schemes suffered from limited error correcting capacity or short keys. In Fuzzy vault schemes the data is stored in the open between chaff points, and this also causes an overhead in storage space. Some quantization schemes were used by Sutcu et al. [23, 24] to generate somewhat stable keys. There were also several works that combine the face data with user specific keys. These include combination with a password by Chen and Chandran [4], user specific token binding by Ngo et al. [17, 28, 29], biometric salting by Savvides et al. [20], and user specific random projection schemes by Teoh and Yuang [27] and Kim and Toh [13]. Although the use of user-specific tokens boosts matching performance while enabling high template security, there are questions raised regarding the protection of the tokens themselves and the contribution of the biometric data towards the performance. Hybrid approaches that combine transform based cancelability with cryptosystem-based security like [8] have also been proposed but give out user specific information to generate the template, creating openings for masquerade attacks.

11.5 Deep Learning for Face Template Protection

In the last few years deep convolutional neural network (CNN) based algorithms like Facenet [21] and Deepface [25] have shown exceptional performance and hold the current state-of-the-art results for face recognition. The key reason behind the success of deep neural networks is their ability to learn representations suited to the task, as opposed to using hand-crafted features. The large availability of labeled data for faces coupled with recent advances in deep CNN architectures and training algorithms has made it possible to learn representations that far exceed the performance of traditional features for face recognition. Consequently biometric authentication using faces has also benefited from this and matching numbers have reached all time highs.

This representation learning ability of neural networks can also be used to learn features suited to template security in addition to discriminative ability. As discussed earlier, cryptographic hash based template security is theoretically ideal but achieving it without some form of information leakage or severe loss in matching performance is challenging. The next two sections discuss two recent approaches to face template security that utilize recent advances in deep learning to address the challenges related to achieving cryptographic hash based security.

11.6 Local Region Hashing

To the best of our knowledge, local region hashing [19] is one of the first approaches that studies the suitability of learned representations for the purpose of cryptographic hash based face template security. The algorithm seeks to achieve exact matching between features extracted from local regions of the face instead of the entire face. Since these regions are smaller, features extracted from them would arguably show lesser variations as compared to features extracted from the entire face. Let us discuss the algorithm and accompanying experiments in further detail.

11.6.1 Algorithm

This approach for template generation consists of the following. First, the face image is divided into a set of selected local regions. Next, features are extracted from each of them and the feature vectors are quantized to eliminate minor variations. Finally, the quantized feature vectors are hashed and stored. Thus, the protected face template consists of a set of hashed features. An overview of the algorithm is shown in Fig. 11.2.

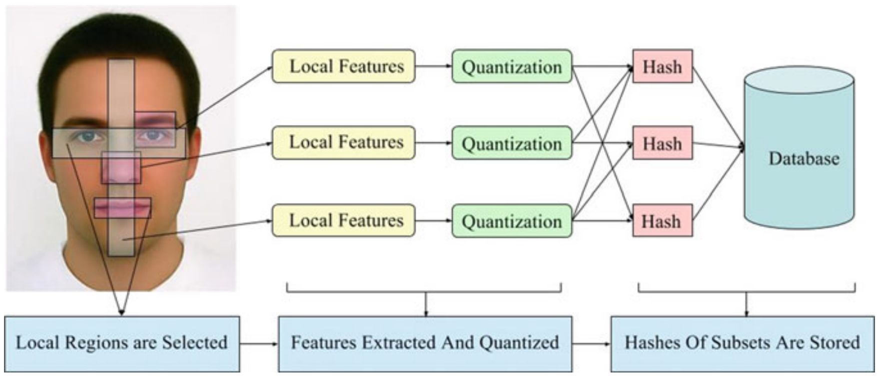


Fig. 11.2 Overview of the local region hashing

11.6.1.1 Facial Region Selection

The purpose of extracting features from fixed local regions of the face is twofold. First, this increases the chances of obtaining exact matches between features. Even though there are inherent variations in holistic features extracted from face samples belonging to different sessions, there exist local regions of the face that remain relatively unchanged, and thus yield features that show much smaller variations. Second, choosing fixed regions, along with tweaks to the design of the framework, ensures that features from a particular region are matched only to features from the same region. This imposes a spatial constraint on the features and reduces the chances of false positive resulting from features from one region being matched to those from others. Thus, the total region of interest is given by a union of V smaller local regions, r, as,

$$R_{interest} = \bigcup_{i=0}^{V-1} r_i \tag{11.1}$$

Desirable characteristics of the selected local regions are: (i) They should yield features that are discriminative. (ii) They should yield features that have higher chances of exact matches. At the pixel level this can be controlled by varying the size and location of the regions. $R_{interest}$ should be such that it spans either the entire face or at least the areas of higher statistical significance for discriminative power. The size of each r_i on the other hand affects the ability to yield features that match exactly. If the size is too large, the features extracted from them may show high variations and if the size is too small, the features extracted may not be discriminative enough. Thus, this can be seen as the first level of control of a two stage process which is geared towards yielding features suited to our task.

For the purpose of experimentation $R_{interest}$ is chosen as the entire face, and equally sized nonoverlapping blocks as the local regions. Hence for a face image of size $(d \times d)$, division into b nonoverlapping blocks would yield local regions of size $(d/b \times d/b)$.

11.6.1.2 Feature Extraction

The second level of control on the representation of the template comes in the form of high-level feature extractors applied to the low-level pixel regions. The higher level features that are used should possess the following characteristics: (i) They should be stable to minor variations and thus be appropriate for exact matching. (ii) They should be discriminative enough to distinguish users and thus have low FAR/FRR. (iii) They should be of sufficient length in order for their hashed versions to be resistant towards brute force attacks, and possess sufficient entropy to be resistant to smarter attacks. For now, the simpler requirements of stability and length are discussed, leaving the more involved entropy-related aspects for the security analysis section. The first feature extraction strategy uses standard feature extractors which have been shown to be useful for face recognition. The second explores the use of a denoising autoencoder to learn a feature representation.

Standard feature extractors

The suitability of two standard feature extractors, namely, histogram of gradients (HoG) [5] and local binary pattern (LBP) histograms [18] is explored. The choice is driven by the motivation that these are the two most basic gradient and texture-based feature extractors and studying their performance for the task at hand would give us valuable insight into the kind of representations suitable for security as well as accuracy. The controllable parameters for HoG features are the number of cells per block, the number of pixels per cell, and the number of orientations. The number of pixels per cell and number of orientations would affect the feature length. Let o denote the number of orientations and $c = m/p$ denote the number of cells if $(p \times p)$ pixels per cell for a region of size $(m \times m)$ are chosen. The final feature length is given by $l_{feat} = o \times c$. For LBP histograms, the variable parameters are the radius and the number of neighbors. The radius affects the resolution of the LBP codes calculated while the number of neighbors determines the feature length. Thus, for LBP histograms calculated with n neighbors per pixel, the feature length is given by $l_{feat} = 2^n$.

The length of the feature vector not only affects the discriminative ability of the features but also the resilience of their hashed versions to brute force attacks. The effect of feature length of discriminative ability and template security are discussed in further detail in the experiments section.

Learned representation

The standard feature extractors are compared to a representation learned from the data itself. Not only does this generate features that are suited to the choice of local regions, but it also enables greater control on the desirable properties of the features. A stacked denoising autoencoder (SdA) [31] is used to learn this representation. The autoencoder is a neural network that maps the input back to itself through some hidden states. The loss between the reconstructed input and the original one is minimized to learn a representation of the size of the number of hidden states. In order to learn a robust representation and prevent the autoencoder from learning the identity function, a denoising autoencoder (dA) tries to reconstruct the input from a corrupted version of it. Thus, the input layer of the neural network is the data, x whose representation we wish to learn. The hidden layer consists of h nodes to which the input layer is fully connected via a set of weights, W along with some bias, b. The output of the hidden layer is a latent representation given by $y = s(Wx + b)$, where s is a nonlinearity such as a sigmoid function. The latent representation is mapped back to the reconstruction z given by $z = s(W'y + b')$. The parameters of this model W, W$'$, b, and b$'$ are optimized to minimize the reconstruction error between z and x. In our case this is given by the squared error, $L(xz) = \parallel x - z \parallel^2$. Multiple dA's can be stacked to form a SdA where the output of each layer is treated as the input to the next and mapped back to the respective layers and finally the reconstructed input.

Gaussian noise is applied to perturb the input during training. Training is done layer wise, in an unsupervised manner, minimizing the reconstruction error for each layer independently. The SdA is trained using the pixel level data from all the blocks seen during enrollment and the trained network is then used as a feature extractor. The feature would correspond to the hidden representation y of the final layer and its length is given by $l_{feat} = h$. In this way, the length of the feature vector can be controlled by varying the number of hidden nodes in the network.

11.6.1.3 Quantization and Combination

After extraction and normalization of each d dimensional feature vector f, such that $f \in [0, 1]^d$, they are quantized by binning each dimension into one of q bins, b of size $1/q$ yielding,

$$f_q \in \left\{b_0, b_1, \ldots, b_{q-1}\right\}^d \tag{11.2}$$

This is done to eliminate minor variations between vectors and increase chances of an exact match. The discriminative power and entropy of the features are also reduced during this step and the effects of this are analyzed in the experiments and security analysis sections. In addition to binning, a region identifier is appended to the feature vector to tag it with the region it was extracted from. Thus, if feature vector $f_q \in \left\{b_1, b_2, \ldots, b_q\right\}^d$ was extracted from local region r_i, the region number i would be appended to f, yielding,

$$f_{qr} \in \left\{b_0, b_1, \ldots, b_{q-1}\right\}^d \parallel i \tag{11.3}$$

This ensures that features would only match to others from the same region and imposes a spatial constraint on local matching.

Another way in which length can be increased for feature vectors is by using combinations of vectors from different regions. This would prove useful in scenarios where low length features are discriminative enough to yield good matching accuracy but not of sufficient length to be resilient to brute force attacks on their hashed versions. Given a set of features $F = \left\{f_{qr0}, f_{qr1}, \ldots, f_{qrV-1}\right\}$, of dimensionality d, and an assumption that, on average, at least k features from different regions match, indexing of all possible combinations of size k is possible. Now, each new feature f'_{qr} would be given by $\|_{j=0}^{k-1} f_{qrj}$ with a new dimensionality of $k \times d$. The feature set for a sample would then be $F' = \left\{f'_{qr0}, f'_{qr1}, \ldots, f'_{qr\binom{V-1}{k}}\right\}$. Thus, there is no loss of exact matching, gain in feature length and intuitively, a reduction in the chances of false positives.

11.6.1.4 Template Protection and Indexing

Given a set of features, F, per sample, the next objective is to hashing and store them in the form of a biometric template. A standard cryptographic hash function $h(x)$ (SHA-256 in this case) is used for hashing each feature in F. This yields a new set T, corresponding to the protected template for one sample, and given by,

$$T = \left\{h(f_{qr0}), h(f_{qr1}), \ldots, h(f_{qrV-1})\right\} \tag{11.4}$$

Thus, the final protected template is a set of hashed local features extracted from the face. If multiple images are enrolled, the template would be a set consisting of hash digests of all the non-repeated quantized feature vectors seen during enrollment.

During authentication, similar local features are extracted from the face image, quantized, and hashed yielding a set of V local features, T_{test}. In verification mode, the template corresponding to the user, $T_{enrolled}$ is retrieved and the matching score is given by the set intersection between the two templates as,

$$score = T_{test} \cap T_{enrolled} \tag{11.5}$$

The index can also be set up for identification where a given sample does not claim to be from any class. In this case, the templates for all users would be gone through and the sample would be identified as the user with the template which showed maximum matches.

11.6.2 Experiments

The framework is tested on the Multi-PIE [10] database using ROC curves and the EER as evaluation metrics. Frontal poses with neutral expressions of the 167 face classes that are common between session 1 and session 2 are used for evaluation. Ten randomly selected images of each user from session 1 are used for enrollment, and all the images from session 2 are used for verification. Thus, the test bed contains variations in lighting and inherent variations in samples taken from different sessions. These variations are arguably sufficient for deployment of the algorithm in scenarios of user compliance. Due to the strict requirements of exact matching, proper alignment of the images is crucial and handled by aligning eye centers for each image. Other specifics including the parameters and implementation libraries used for the experiments are described below.

Each face image is aligned to have their eyes positioned at roughly the same location and cropped to remove most of the background yielding a final face image of size 200×200. The face image is then divided into $V = 100$ local regions in the form of nonoverlapping blocks of size 20×20 with $R_{interest}$ from Eq. 11.1 being the entire image. Next, features f are extracted from each block followed by binning each dimension into $q = 4$ bins and appending the region identifier $i \in \{1, 2, \ldots, 100\}$

yielding f_{qr} from Eq. 11.3. For HoG features the number of cells are fixed to 4 and the number of orientations, o, are varied. For LBP features the number of neighbors, n, are varied. For the SdA, two layers are used, fixing the number of hidden states in the first layer to 576 and varying the number of hidden layers in the second hidden layer to the desired dimensionality of the learned representation. Zero mean Gaussian noise with 0.5 variance is used as noise, and training is done for 20 epochs for each layer, with a learning rate of 0.3. Note that blocks extracted from the enrollment set are used to train the autoencoder for simplicity but since we merely want to learn a feature representation for faces, the training could have been done on an entirely different database as well. Scikit-image is used for calculating HoG and LBP features. The autoencoder is implemented using Theano [2, 3]. Experimentation is done with feature lengths ranging from 8–32 for HoG, 16–256 for LBP and 16–256 for the autoencoder. Next each f_{qr} is hashed using SHA-256 as the hashing function h, yielding the set T from Eq. 11.4. During enrollment a set of non-repeating features is built for each user. For verification, the set intersection between the set generated from the test sample, T_{test_i} and the set enrolled for the claimed user, T_{user_i} is computed. This yields the $score$ from Eq. 11.5 as $T_{test_i} \cap T_{user_i}$. The score ranges from 0–100 in this case. Genuine scores are computed by comparing test samples against their true classes while impostor scores are computed against each of the other enrolled classes.

Figure 11.3 shows the ROC curves for HoG, LBP, and autoencoder features respectively (note that some of the curves are not smooth due to the discrete nature of the score and overlaps at varying thresholds). It can be seen that HoG features perform very poorly showing a reasonable EER of 15.3% only at a very low bits of security of 8, making it unsuitable for security. LBP features perform better with an EER of 21.3% at six neighbors or 128 bits of security. The feature representation learned by the SdA shows the best results at an acceptable 128 bits of security with an EER of 16.6%. Interestingly, the EER is not far behind even at 256 bits. It is worth noting that the matching score in all the cases was never more than 60–70% of the total number of regions, confirming that such a hashing-based security scheme would be infeasible if holistic features from the entire face were used.

11.6.3 Security Analysis

The security of the algorithm is studied in a stolen template scenario, where in, if an attacker gains access to the secure templates, it should be infeasible to extract the original biometric data from them. It is assumed that the hash function, SHA-256, is collision resistant and follows the random oracle model in which the hashed digests reveal no dependencies between them. In such a scenario, the only way to attack the system is by guessing values in the feature domain, hashing them and comparing them to the stored hashes in the database. A naive attacker would assume uniformity in the feature space and search through every possible value. Hence, in this case the bits of security offered by the system is proportional to the size of the

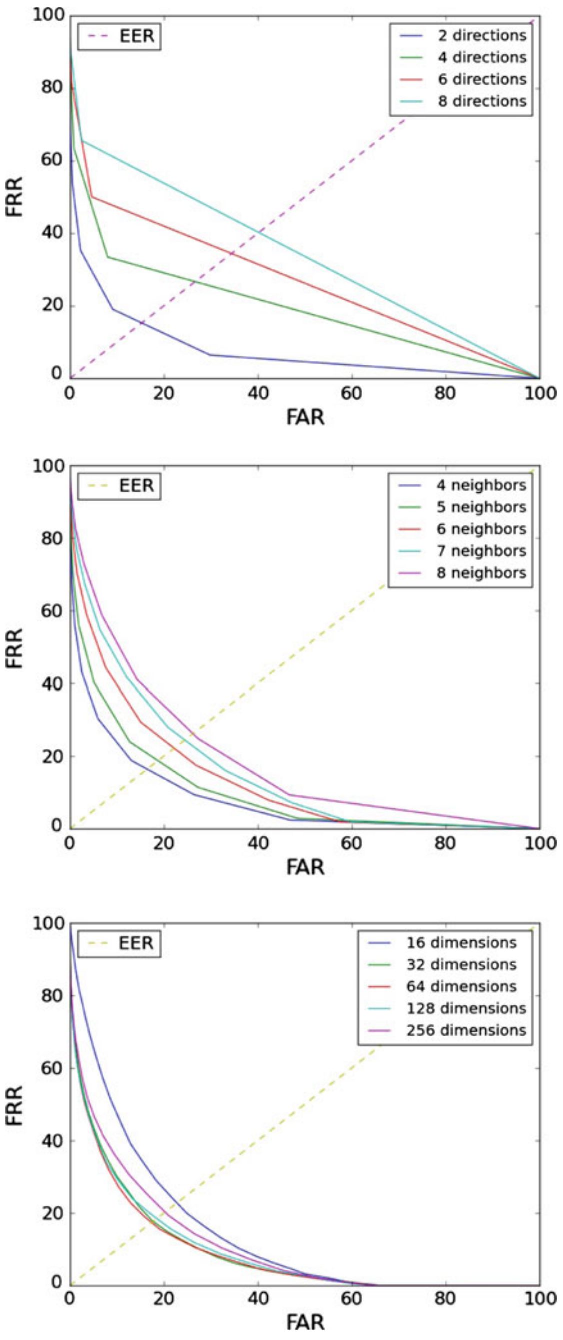


Fig. 11.3 ROC curves for HoG (*top*), LBP (*mid*) autoencoder (*bottom*) features

Table 11.1 Feature security

Feature extractor	$q^{l_{feat}}$	$MI(Db_{en}, Db_{at})$
HoG	2^{64}	0.1610
LBP	2^{64}	0.1068
	2^{128}	0.0962
	2^{256}	0.1029
SdA	2^{64}	0.0739
	2^{128}	0.0870
	2^{256}	0.1020

search space, given by $T \times q^{l_{feat}}$, where T is the selected score threshold. In reality, the feature space is far from uniform and approximating its true distribution in the lack of sufficient data is a daunting task. Instead, it is somewhat easier to quantify the amount by which a smart attacker can reduce the search space from $q^{l_{feat}}$. It is assumed that the attacker has knowledge of the algorithm, the feature extractor used, and access to face data, Db_{at} that is not enrolled in the system. The amount by which an attacker can reduce his search space using information leakage, is quantified in the form of mutual information (MI), between the features extracted from Db_{at} and the enrolled database, Db_{en}. It is hypothesized that due to the highly multimodal nature of the data from different regions and face classes, a smart attack will have to be made region wise and thus, the information leakage is reported in terms of the average MI between regions from the two databases.

$$MI(Db_{en}, Db_{at}) = \frac{(\sum_{i=1}^{V} MI(F_{qr_i}^{en}, F_{qr_i}^{at})}{V} \tag{11.6}$$

where F_{qr_i} is the set of quantized features extracted from region i.

For the purpose of this experiment Db_{en} is a set of 20 randomly chosen face classes while Db_{at} is a distinct set of rest of the 147 classes not in Db_{en}. The MI is calculated using the non-parametric entropy estimation toolkit (NPEET) which implements the MI estimator proposed by Kraskov et al. [14]. Since the non-parametric estimation is specific to the data distribution, the MI values are best seen as a comparative measure between the feature extractors, and not on a universal scale. The MI values for different feature extractors at varying feature lengths is shown in Table 11.1, and further details about the values themselves can be found in [14].

11.7 Deep Secure Encoding

Let us now discuss an alternative approach that addresses the shortcomings of local region hashing in terms of template security as well as matching accuracy. The algorithm uses a deep convolutional neural network (CNN) to learn a robust mapping of face classes to codes referred to as maximum entropy binary (MEB) codes. The

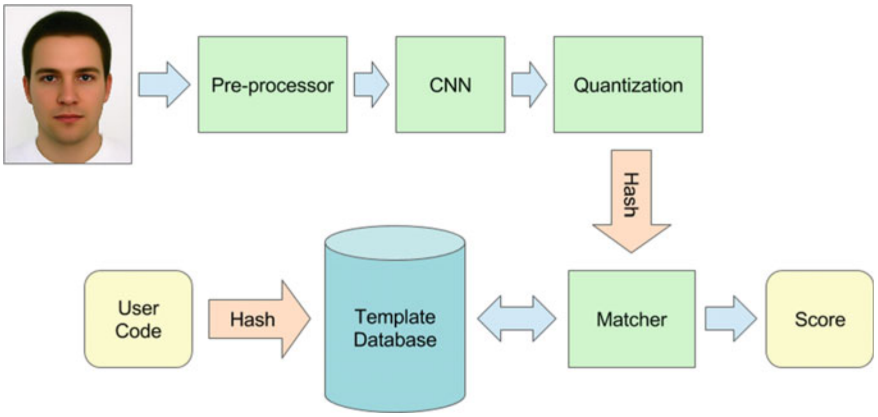


Fig. 11.4 Overview of deep secure encoding

mapping is robust enough to tackle the problem of exact matching, yielding the same code for new samples of a user as the code assigned during training. This exact matching makes it possible to store the cryptographic hash of the code as the template for the user. Once hashed, the template has no correlation with the code assigned to the user. Furthermore, the codes assigned to users are bit-wise randomly generated and thus, possess maximum entropy, and have no correlation with the original biometric modality (the user's face). These properties make attacks on the template very difficult, leaving brute force attacks in the code domain and complex dictionary attacks in the input domain as the only feasible options. Cancelability can also be easily achieved by changing the codes assigned to users and relearning the mapping.

11.7.1 Algorithm

We now describe the individual components of our architecture in more detail. An overview of the algorithm is shown in Fig. 11.4.

11.7.1.1 Convolutional Neural Networks

Convolutional neural networks (CNNs) [15] are biologically inspired models, which contain three basic components: convolution, pooling, and fully connected layers. In the convolution layer one tries to learn a filter bank given input feature maps. The input of a convolution layer is a 3D tensor with d number of 2D feature maps of size $n_1 \times n_2$. Let x_{ijk} denote the component at row j and column k in the ith feature map, and $x_i^{(l)}$ denote the complete ith feature map at layer l. If one wants to learn h_f set of filters of size $f_1 \times f_2$, the output $x^{(l+1)}$ for the next layer will still be a 3D tensor with h_f number of 2D feature maps of size $(n_1 - f_1 + 1) \times (n_2 - f_2 + 1)$. More formally, the convolution layer computes the following:

$$x_j^{(l+1)} = s\left(\sum_i F_{ij}^{(l)} * x_i^{(l)} + b_j^{(l)}\right) \tag{11.7}$$

where $F_{ij}^{(l)}$ denotes the filter that connects feature map x_i to output map $x_j^{(l)}$ at layer l, $b_j^{(l)}$ is the bias for the jth output feature map, $s(\cdot)$ is some element-wise nonlinearity function and $*$ denotes the discrete 2D convolution.

The pooling (or subsample) layer takes a 3D feature map and tries to downsample/summarize the content to lesser spatial resolution. Pooling is commonly done for every feature map independently and with nonoverlapping windows. The intuition behind such an operation is to have some built in invariance against small translations. Additionally this reduces the spatial resolution and thus saves computation for the upper layers. For average (mean) pooling, the output will be the average value inside the pooling window, and for max pooling the output will be the maximum value inside the pooling window.

The fully connected layer connects all the input units from a lower layer l to all the output units in the next layer $l + 1$. In more detail, the next layer output is calculated by

$$x^{(l+1)} = s(W^{(l)} x^{(l)} + b^{(l)}) \tag{11.8}$$

where $x^{(l)}$ is the vectorized input from layer l, $W^{(l)}$ and $b^{(l)}$ are the parameters of the fully connected layers at layer l.

A CNN is commonly composed of several stacks of convolution and pooling layers followed by a few fully connected layers. The last layer is normally associated with some loss to provide training signals, and the training for CNN can be done by doing gradient descent on the parameters with respect to the loss. For example, in classification the last layer is normally a softmax layer and cross-entropy loss is calculated against the 1 of K representation of the class labels. In more detail, let $x^{(L)} = Wx^{(L-1)} + b$ be the pre-activation of the last layer, $\mathbf{t}$ denotes the final output and t_k the kth component of $\mathbf{t}$, and $\mathbf{y}$ denote the target 1 of K vector and y_k the kth dimension of that vector, then

$$t_k = \frac{\exp\{x_k^{(L)}\}}{\sum_j \exp\{x_j^{(L)}\}} \tag{11.9}$$

$$L(\mathbf{t}, \mathbf{y}) = \sum_j y_j \log t_j \tag{11.10}$$

where L is the loss function.

11.7.1.2 Maximum Entropy Binary Codes

The first step of training is to assign unique codes to each user to be enrolled. Note that these codes are internally used for training during enrollment, and are not supplied to the user or retained in an unprotected form after training. From a template security point of view, these codes should ideally possess two properties. First, they should posses high entropy. Since a hash of these codes is the final protected template, the higher the entropy of the codes, the larger the search space for a brute force attack would be. In order to make brute force attacks in the code domain infeasible, we use binary codes with a minimum dimensionality $K = 256$ and experiment with values up to $K = 1024$. The second desirable property of the codes is that they should not be correlated with the original biometric modality. Any correlation between the biometric samples and the secure codes can be exploited by an attacker to reduce the search space during a brute force attack. One example to illustrate this can be to think of binary features extracted from faces. Even though the dimensionality of the feature vector may be high, given the feature extraction algorithm and type of data, the number of possible values the vector can take is severely reduced. In order to prevent such reduction of entropy, the codes used are bit-wise randomly generated and have no correlation with the original biometric samples. This makes the space to be hashed uniformly distributed. More precisely, let $c_i \sim \mathbf{B}(1, 0.5)$ be the binary variable for each bit of the code, where $\mathbf{B}(1, 0.5)$ is the maximum entropy Bernoulli distribution, and the resultant MEB code with K independent bits is thus $\mathbf{C} = [c_1, c_2, \ldots, c_K]$. The code for user u is denoted by $\mathbf{C}_u$.

11.7.1.3 Learning the Mapping

In order to learn a robust mapping of a user's face samples to the codes, some modifications are made to the standard CNN training procedure. As shown in Fig. 11.5, the 1 of K encoding of the class labels is replaced by the MEB codes $\mathbf{C}_u$ assigned to each user. Since several bits of the network output are going to be one instead of a single bit, sigmoid activation is used instead of softmax. In more detail

$$t_k = \frac{1}{1 + \exp\{-x_j^{(L)}\}} \tag{11.11}$$

$$L(\mathbf{t}, \mathbf{C}) = \sum_j \{c_j \log t_j + (1 - c_j) \log(1 - t_j)\} \tag{11.12}$$

where t_k is the kth output from the last layer and L is the binary cross-entropy loss.

Data Augmentation

Deep learning algorithms generally require a large number of training samples whereas, training samples are generally limited in the case of biometric data. In order to magnify the number of training samples per user, the following data

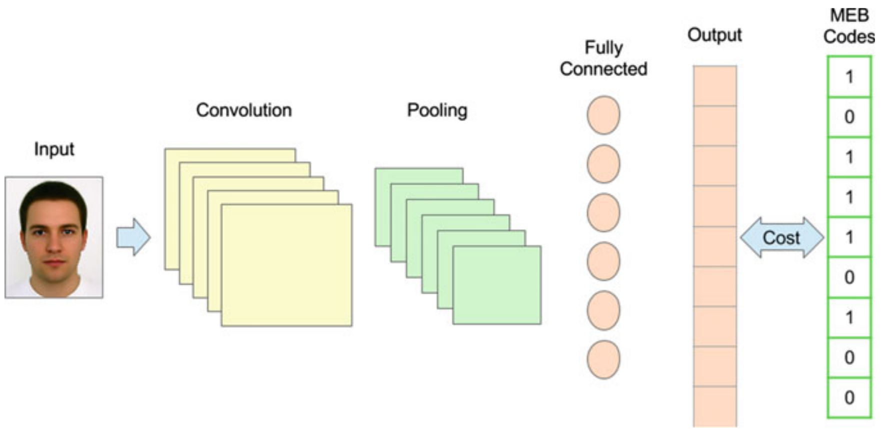


Fig. 11.5 Learning the mapping

augmentation is performed. For each training sample of size $m \times m$ all possible crops of size $n \times n$ are extracted. Each crop is also flipped along its vertical axis yielding a total of $2 \times (m - n + 1) \times (m - n + 1)$ crops. The crops are then resized back to $m \times m$ and used for training the CNN.

Regularization

The large learning capacity of deep neural networks comes with the inherent risk of overfitting. The number of parameters in the network are often enough to memorize the entire training set, and the performance of such a network does not generalize to new data. In addition to general concerns, mapping to MEB codes is equivalent to learning a highly complex function, where each dimension of the function output can be regarded as an arbitrary binary partition of the classes. This further increases the risk of overfitting and powerful regularization techniques need be employed to achieve good matching performance.

Dropout [11] is applied to all fully connected layers with 0.5 probability of discarding a hidden activation. Dropout is a very effective regularizer and can also be regarded as training an ensemble of an exponential number of neural networks that share the same parameters, therefore reducing the variance of the resulting model.

11.7.1.4 Protected Template

Even though MEB codes assigned to each user have no correlation with the original samples, another step of taking a hash of the code is required to generate the protected template. Given the parameters of the network, it is not possible to entirely recover the original samples from the code (due to the max pooling operation in the forward pass of the network) but, some information is leaked. Using a hash digest of the code as the final protected template prevents any information leakage. The hash function used can be any function that follows the random oracle model. For experimentation SHA-512 is utilized, yielding the final protected template $\mathbf{T}_u = \text{SHA512}(\mathbf{C}_u)$.

During verification, a new sample of the enrolled user is fed through the network to get the network output $\mathbf{y}_{out} = \mathbf{t}$. This output is then binarized via a simple thresholding operation yielding the code for the sample $\mathbf{s}_{out} = [s_1, s_2, \ldots, s_K]$, where $s_i = \mathbf{1}(t_i > 0.5)$ and $\mathbf{1}(\cdot)$ is the indicator function. At this point, the SHA-512 hash of the code, $\mathbf{H}_{out} = \text{SHA512}(\mathbf{s}_{out})$ could be taken and compared with the stored hash $\mathbf{T}_u$ for the user. Due to the exact matching nature of the framework, this would yield a matching score of true/false nature. This is not ideal for a biometric-based authentication system since it is desirable to obtain a tunable score in order to adjust the false accept (FAR) and false reject rates (FRR). In order to obtain an adjustable score, several crops and their flipped counterparts are taken for the new sample (in the manner described in Sect. 11.7.1.3) and $\mathbf{H}_{out}$ is calculated for each one, yielding a set of hashes $\mathbb{H}$. The final matching score is defined as the number of $\mathbf{H}_{outs}$ in $\mathbb{H}$ that match the stored template, scaled by the cardinality of $\mathbb{H}$. Thus, the score for matching against user u is given by,

$$score = \frac{\sum_{\mathbf{H}_i \in \mathbb{H}} \mathbf{1}(\mathbf{H}_i = \mathbf{T}_u)}{|\mathbb{H}|} \tag{11.13}$$

Now the score can be set to achieve the desired value of FAR/FRR. Note that, the framework provides the flexibility to work in both verification and identification modes. For identification $\mathbb{H}$ can be matched against templates of all the users stored in the database.

11.7.2 Experiments

We now discuss the databases, evaluation protocols, and specifics of the parameters used for experimental evaluation.

11.7.2.1 Databases

This study tackles the problem of using faces as passwords and thus, utilizes face databases that have been collected in controlled environments for experimentation. The evaluation protocols include variations in lighting, session, and pose that would be typical to applications like face unlock since a reasonable degree of user compliance is expected.

The CMU PIE [22] database consists of 41,368 images of 68 people under 13 different poses, 43 different illumination conditions, and with four different expressions. Five poses (c27, c05, c29, c09 and c07) and all illumination variations are used for experimentation. 10 images are randomly chosen for training and the rest are used for testing.

The extended Yale Face Database B [9] contains 2432 images of 38 subjects with frontal pose and under different illumination variations. The cropped version of the database is used for experimentation. Again, 10 randomly selected images are used for training and the rest for testing.

The CMU Multi-PIE [10] face database contains more than 750,000 images of 337 people recorded in four different sessions, 15 view points and 19 illumination conditions. This database is used to highlight the algorithm's robustness to changes in session and lighting conditions. Two sessions (3 and 4) which have the most number of common users (198) between them are chosen. Ten randomly chosen frontal faces from session 3 are used for enrollment and all frontal faces from session 4 are used for testing.

11.7.2.2 Evaluation Metrics

Genuine accept rate (GAR) at 0 false accept rate (FAR) is used as the evaluation metric. The equal error rate (EER) is also reported as an alternative operating point for the system. Since the train-test splits used are randomly generated, the mean and standard deviation of the results for 10 different random splits are reported.

11.7.2.3 Experimental Parameters

The same training procedure is used for all databases. The CNN architecture used is as follows: two convolutional layers of 32 filters of size 7×7 and 64 filters of size 7×7, each followed by max pooling layers of size 2×2. The convolutional and pooling layers are followed by two fully connected layers of size 2000 each, and finally the output. Rectifier activation function $s(x) = \max(x, 0)$ is used for all layers, and dropout is applied with 0.5 probability of discarding activations to both fully connected layers.

MEB codes of dimensionality $K = 256, 1024$ are assigned to each user. All training images are resized to $m \times m = 64 \times 64$ and roughly aligned using eye center locations. For augmentation $n \times n = 57 \times 57$ crops are used yielding 64 crops per image. Each crop is also illumination normalized using the algorithm in [26]. The network is trained by minimizing the cross-entropy loss against user codes for 20 epochs using mini-batch stochastic gradient descent with a batch size of 200. Five of the training samples are initially used for validation to determine the mentioned training parameters. Once the network is trained, the SHA-512 hashes of the codes are stored as the protected templates and the original codes are purged. During verification, crops are extracted from the new sample, preprocessed, and fed through the trained network. Finally, the SHA-512 hash of each crop is calculated and matched to the stored template, yielding the matching score in Eq. 11.3.

Table 11.2 Verification results obtained from various datasets

Database	K	GAR@0FAR (%)	EER (%)
PIE	256	93.22 ± 2.61	1.39 ± 0.20
	1024	90.13 ± 4.30	1.14 ± 0.14
Yale	256	96.74 ± 1.35	0.93 ± 0.18
	1024	96.49 ± 2.30	0.71 ± 0.17
Multi-PIE	256	95.93 ± 0.55	1.92 ± 0.27
	1024	97.12 ± 0.45	0.90 ± 0.13

11.7.2.4 Results

The results of our experiments are shown in Table 11.2. The mean and standard deviation of GAR at zero FAR, and EER are reported for the 10 different train-test splits at code dimensions, $K = 256, 1024$. GARs up to ~90% on PIE, ~96% on Yale, and ~97% on Multi-PIE with up to $K = 1024$ are achieved at the strict operating point of zero FAR. During experimentation it was observed that the results were stable with respect to K, making the parameter selectable purely on the basis of desired template security. In order to get an idea of the system performance with respect to the operating point, the GAR and FAR of the system are shown with respect to the matching score for $K = 256$ in Fig. 11.6. It is noteworthy that the system has very low FAR values even at low matching scores due to the strict exact matching requirements.

A comparison of the results to other face template protection algorithms on the PIE database is shown in Table 11.3. GAR at an FAR of 1% is compared as this is the reported operating point in [7]. For security level, the code dimensionality parameter (K) is compared to the equivalent parameter in the shown approaches. In the absence of algorithm parameters, this is a good measure of the security level against brute force attacks.

11.7.3 Security Analysis

Once again, the security of the system is analyzed in a stolen template scenario. It is assumed that the attacker has possession of the templates, and knowledge of the template generation algorithm. Given the templates, the attacker's goal is to extract information about the original biometric of the users. Since the hash function used to generate the templates is a one way transformation function, no information about the MEB codes can be extracted from the protected templates. Thus, the only way in which the attacker can get the codes is by brute forcing through all possible values the codes can take, hash each one, and compare them to the templates. The search space for such brute force attacks is now analyzed.

Fig. 11.6 GAR and FAR with respect to matching score at K = 256 for PIE (*top*), Yale (*mid*) and Multi-PIE (*bottom*) databases

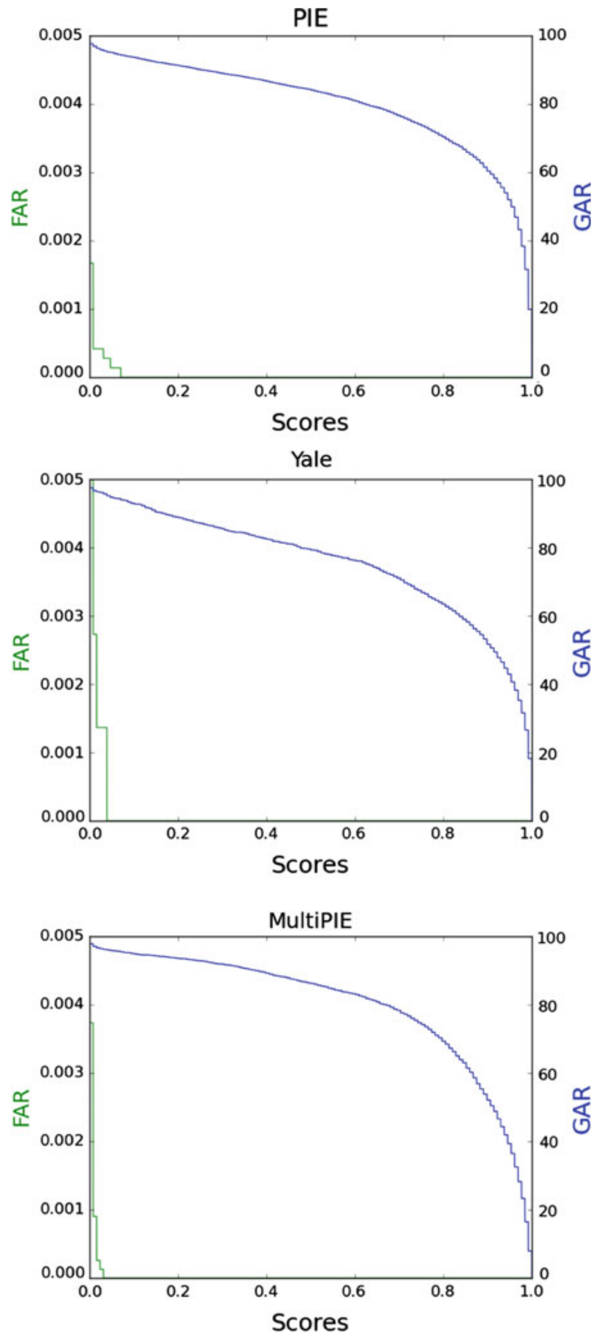


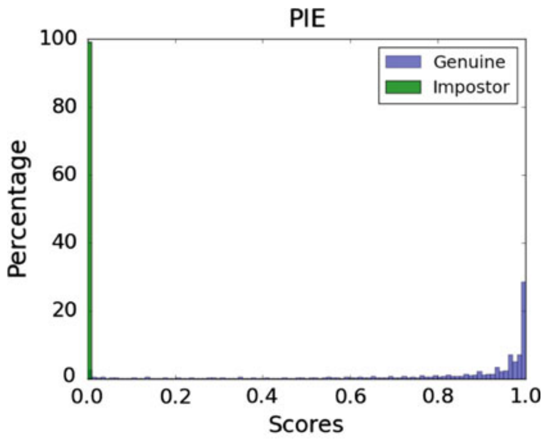
Table 11.3 Performance comparison with other algorithms on PIE dataset

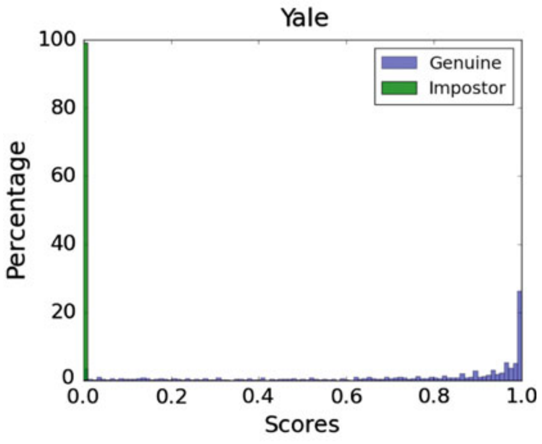
Method	K	GAR@1FAR (%)	EER (%)
Hybrid approach [8]	210	90.61	6.81
BDA [7]	76	96.38	–
MEB encoding	**1024**	**97.59**	**1.14**

In the absence of the CNN parameters, the search space for brute force attacks would be 2^K where K is the number of dimensions of the MEB code. This is because the MEB codes are bit-wise randomly generated and uncorrelated to the original biometric data. Since a minimum of $k = 256$ is used, the search space would be of the order of 2^{256} or larger, making brute force attacks computationally infeasible.

An attack given the CNN parameters is now analyzed. With the CNN parameters, it would make sense to generate attacks in the input domain of the network and try to exploit the FAR of the system. Brute forcing through all possible values in the input domain would yield a search space much larger than 2^K. Thus, in a practical scenario, attackers would most likely perform a dictionary attack using a large set of faces that is available to them. Even though it is not straightforward to analyze the reduction of the attacker's search space due to the knowledge of the system parameters, the FAR of the system under the aforementioned attack scenario is arguably a good indicator of the template security. The genuine and imposter score distributions when all other users other than the genuine are treated as imposter are shown in Fig. 11.7. It can be seen that the imposter scores are mostly zero, indicating that there are very few false accepts in this scenario. The genuine and imposter distributions under a dictionary attack using an attacker database consisting of all frontal images of the Multi-PIE database and genuine database consisting of the smaller Yale database is shown in Fig. 11.8. Again, it can be seen that there are minimal false accepts indicating that the model does not easily accept external faces even when they are preprocessed in the same manner as the enrolled ones. Separately, a large number of random noise samples are also used as an attack to verify that the CNN does not trivially learn how to map large portions of the input space to the learned codes. In this case, there are no false accepts showing that the model is indeed not trivially learning to map all inputs to assigned codes. Hence, even though it is not straightforward to quantify the reduction in the search space of codes due to knowledge of the CNN parameters, it has been empirically shown that false accepts are difficult due to the strict exact matching requirements of the system.

It is worth noting that even if a MEB code is obtained by the attacker, reconstructing the original biometric in an exact manner is not possible due to the pooling and dropout operations in the CNN. Furthermore, knowledge of one code reveals no information about the others since they are uncorrelated. Thus, if a security breach is detected, it is safe to use a new set of MEB codes to generate another set of templates.





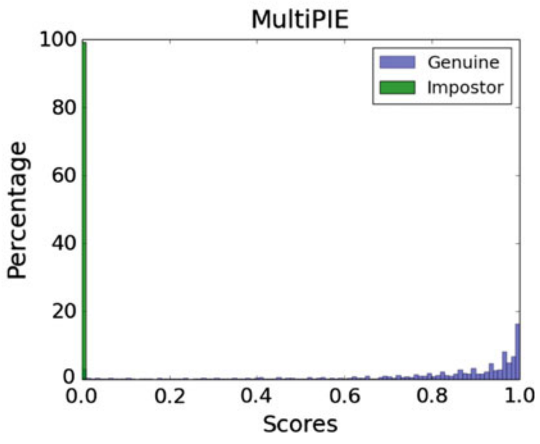


Fig. 11.7 Genuine and imposter distributions from PIE (*top*), Yale (*mid*) and Multi-PIE (*bottom*) databases

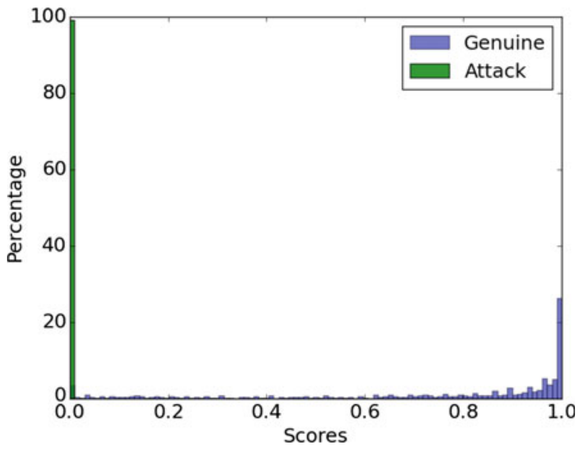


Fig. 11.8 Genuine and imposter distributions under a dictionary attack in the input space

11.8 Future Directions

We have discussed two very different approaches to enabling the use of cryptographic hash functions for face template protection. Local region hashing highlights the challenges associated with cryptographic hash based template security for face biometrics, and confirms the intuition regarding the need for learning representations suitable for security in addition to matching. Even though the algorithm offers a higher standard of template security, it suffers in terms of matching accuracy and uniformity of the hashed space. On the other hand, deep secure encoding takes a different approach by designing the ideal representation for template security first, and then learning a mapping to it. In the process, the algorithm eliminates all the downfalls of local region hashing as well and most previous approaches to face template protection. Even though deep secure encoding achieves state-of-the-art template security as well as matching accuracy, it does not come without its own pitfalls.

One of the disadvantages of deep secure encoding comes to light when we think about enrolling new subjects. Since the mapping to the MEB codes for all enrolled users is learned jointly, the network would need to be retrained in order to enroll new subjects. Deep neural network retraining can be a time consuming task making the process of new enrollment inefficient. That said, neural networks are generally trained by fine tuning a representation that has already been pretrained on relevant data, dramatically reducing the training time. In the context of deep secure encoding, a base network that has been trained on faces in general might be used for initialization. Thus, fine tuning on the classes to be enrolled would be considerably less time consuming. There is, however, a bigger issue with the need for retraining. Since the system does not retain the original face data, it would have to request samples from all the previous users in order to retrain the network. This may not always be feasible. One can think of some workarounds like maintaining different networks for different sets of users and adding a new network when a new set of users needs to be enrolled, but they could still be complicated to practically deploy.

A possible solution to the above-mentioned issues could be to use a deep neural network to learn a universal locally sensitive hashing function for faces. Instead of learning a mapping to fixed codes like deep secure encoding, a binary code can be learned using a loss function that minimizes the distance between codes for the same user and maximizes the distance between codes for different users. One important detail to keep in mind during the implementation of such an approach is that the smoothness of the binary representation space can be exploited to perform hill climbing attacks on the learned codes. Needless to say, this approach would also require a much larger data pool for training. Whether this would work in a practice or not, is yet to be seen. Other improvements to deep secure encoding include pre-training before enrollment and possible combination with local region hashing to learn mappings for smaller facial regions.

In conclusion, deep learning based algorithms show significant potential for learning representations that are suited to biometric template protection. Although not perfect, algorithms like deep secure encoding achieve impressive results and pave the way to towards high standards of template security with minimal loss in matching accuracy. Furthermore, the generic nature of the presented algorithms enable their use for any task where neural networks are applicable; potentially motivating privacy-centric research in other domains.

References

1. M. Ao, S.Z. Li, Near infrared face based biometric key binding, in *Advances in Biometrics* (Springer, Berlin, 2009), pp. 376–385
2. F. Bastien, P. Lamblin, R. Pascanu, J. Bergstra, I.J. Goodfellow, A. Bergeron, N. Bouchard, Y. Bengio, Theano: new features and speed improvements, in *Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop* (2012)
3. J. Bergstra, O. Breuleux, F. Bastien, P. Lamblin, R. Pascanu, G. Desjardins, J. Turian, D. Warde-Farley, Y. Bengio, Theano: a CPU and GPU math expression compiler, in *Proceedings of the Python for Scientific Computing Conference (SciPy)* (2010) (Oral Presentation)
4. B. Chen, V. Chandran, Biometric based cryptographic key generation from faces, in *9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications* (IEEE, 2007), pp. 394–401
5. N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005. CVPR 2005*, vol. 1 (IEEE, 2005), pp. 886–893
6. Y. Dodis, L. Reyzin, A. Smith, Fuzzy extractors: how to generate strong keys from biometrics and other noisy data, in *International Conference on the Theory and Applications of Cryptographic Techniques* (Springer, 2004), pp. 523–540
7. Y.C. Feng, P.C. Yuen, Binary discriminant analysis for generating binary face template. IEEE Trans. Inf. Forensics Secur. **7**(2), 613–624 (2012)
8. Y.C. Feng, P.C. Yuen, A.K. Jain, A hybrid approach for generating secure and discriminating face template. IEEE Trans. Inf. Forensics Secur. **5**(1), 103–117 (2010)

9. A.S. Georghiades, P.N. Belhumeur, D.J. Kriegman, From few to many: illumination cone models for face recognition under variable lighting and pose. IEEE Trans. Pattern Anal. Mach. Intell. **23**(6), 643–660 (2001)
10. R. Gross, I. Matthews, J. Cohn, T. Kanade, S. Baker, Multi-pie. Image Vis. Comput. **28**(5), 807–813 (2010)
11. G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors (2012). arXiv:1207.0580
12. A.K. Jain, K. Nandakumar, A. Nagar, Biometric template security. EURASIP J. Adv. Signal Process. **2008**, 113 (2008)
13. Y. Kim, K.-A. Toh, A method to enhance face biometric security, in *First IEEE International Conference on Biometrics: Theory, Applications, and Systems, 2007. BTAS 2007* (IEEE, 2007), pp. 1–6
14. A. Kraskov, H. Stögbauer, P. Grassberger, Estimating mutual information. Phys. Rev. E **69**(6), 066138 (2004)
15. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, in *Proceedings of the IEEE* (1998), pp. 2278–2324
16. H. Lu, K. Martin, F. Bui, K.N. Plataniotis, D. Hatzinakos, Face recognition with biometric encryption for privacy-enhancing self-exclusion, in *2009 16th International Conference on Digital Signal Processing* (IEEE, 2009), pp. 1–8
17. D.C.L. Ngo, A.B.J. Teoh, A. Goh, Biometric hash: high-confidence face recognition. IEEE Trans. Circuits Syst. Video Technol. **16**(6), 771–775 (2006)
18. T. Ojala, M. Pietikainen, T. Maenpaa, Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 971–987 (2002)
19. R.K. Pandey, V. Govindaraju, Secure face template generation via local region hashing, in *2015 International Conference on Biometrics (ICB)* (IEEE, 2015), pp. 1–6
20. M. Savvides, B.V.K. Vijaya Kumar, P.K. Khosla, Cancelable biometric filters for face recognition, in *ICPR 2004*, vol. 3 (IEEE, 2004), pp. 922–925
21. F. Schroff, D. Kalenichenko, J. Philbin, Facenet: a unified embedding for face recognition and clustering, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 815–823
22. T. Sim, S. Baker, M. Bsat, The CMU pose, illumination, and expression (PIE) databas, in *Proceedings. Fifth IEEE International Conference on Automatic Face and Gesture Recognition, 2002* (IEEE, 2002), pp. 46–51
23. Y. Sutcu, H.T. Sencar, N. Memon, A secure biometric authentication scheme based on robust hashing, in *Proceedings of the 7th Workshop on Multimedia and Security* (ACM, 2005), pp. 111–116
24. Y. Sutcu, Q. Li, N. Memon, Protecting biometric templates with sketch: theory and practice. IEEE Trans. Inf. Forensics Secur. **2**(3), 503–512 (2007)
25. Y. Taigman, M. Yang, M. Ranzato, L. Wolf, Deepface: closing the gap to human-level performance in face verification, in *2014 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2014), pp. 1701–1708
26. X. Tan, B. Triggs, Enhanced local texture feature sets for face recognition under difficult lighting conditions. IEEE Trans. Image Process. **19**(6), 1635–1650 (2010)
27. A. Teoh, C.T. Yuang, Cancelable biometrics realization with multispace random projections. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **37**(5), 1096–1106 (2007)
28. A.B.J. Teoh, D.C.L. Ngo, A. Goh, Personalised cryptographic key generation based on face-hashing. Comput. Secur. **23**(7), 606–614 (2004)
29. A.B.J. Teoh, A. Goh, D.C.L. Ngo, Random multispace quantization as an analytic mechanism for biohashing of biometric and random identity inputs. IEEE Trans. Pattern Anal. Mach. Intell. **28**(12), 1892–1901 (2006)
30. M. Van Der Veen, T. Kevenaar, G.-J. Schrijen, T.H. Akkermans, F. Zuo, et al., Face biometrics with renewable templates, in *Proceedings of SPIE*, vol. 6072 (2006), pp. 60720J

31. P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, P.-A. Manzagol, Stacked denoising autoencoders: learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. **11**, 3371–3408 (2010)
32. Y. Wu, B. Qiu, Transforming a pattern identifier into biometric key generators, in *ICME 2010* (IEEE, 2010), pp. 78–82

Chapter 12
Deep Triplet Embedding Representations for Liveness Detection

Federico Pala and Bir Bhanu

Abstract Liveness detection is a fundamental element for all biometric systems that have to be safe against spoofing attacks at sensor level. In particular, for an attacker it is relatively easy to build a fake replica of a legitimate finger and apply it directly to the sensor, thereby fooling the system by declaring its corresponding identity. In order to ensure that the declared identity is genuine and it corresponds to the individual present at the time of capture, the task is usually formulated as a binary classification problem, where a classifier is trained to detect whether the fingerprint at hand is real or an artificial replica. In this chapter, unlike the binary classification model, a metric learning approach based on triplet convolutional networks is proposed. A representation of the fingerprint images is generated, where the distance between feature points reflects how much the real fingerprints are dissimilar from the ones generated artificially. A variant of the triplet objective function is employed, that considers patches taken from two real fingerprint and a replica (or two replicas and a real fingerprint), and gives a high penalty if the distance between the matching couple is greater than the mismatched one. Given a test fingerprint image, its liveness is established by matching its patches against a set of reference genuine and fake patches taken from the training set. The use of small networks along with a patch-based representation allows the system to perform the acquisitions in real time and provide state-of-the-art performances. Experiments are presented on several benchmark datasets for liveness detection including different sensors and fake fingerprint materials. The approach is validated on the cross-sensor and cross-material scenarios, to understand how well it generalizes to new acquisition devices, and how robust it is to new presentation attacks.

F. Pala (✉) · B. Bhanu
Center for Research in Intelligent Systems - University of California, Riverside Winston Chung Hall Suite 216, 900 University Avenue, Riverside, CA 92521-0425, USA
e-mail: fedpala@ucr.edu

B. Bhanu
e-mail: bhanu@cris.ucr.edu

B. Bhanu and A. Kumar (eds.), *Deep Learning for Biometrics*, Advances in Computer Vision and Pattern Recognition,
DOI 10.1007/978-3-319-61657-5_12

12.1 Introduction

Biometrics authentication systems have become part of the daily routine for millions of people around the world. A large number of people use their fingerprints every day in order to pass the security checkpoints at airports, to access their personal mobile devices [1] and to access restricted areas. The popularity of this biometric with respect of others such as face and iris recognition lies on its reliability, that has been proven during the last decades, its implementation at an affordable cost, and especially on the simplicity of touching a surface to get immediately authenticated.

Unfortunately, all these advantages come with various security and privacy issues. Different attacks can be performed to the authentication system in order to grant access to some exclusive area or to steal confidential data. For instance, the software and the network configuration can have security holes or bugs, and the matching algorithms can be fooled if the attacker knows the software implementation details [2]. Moreover, whereas a physical key or badge can be replaced, fingerprints are permanent and the pattern on their surface can be easily captured and reproduced. It can be taken from a high-resolution photograph or from a print left on a surface such as a mug or even a piece of paper. A high-quality reproduction of the pattern on some gummy material can be simply applied to the scanner [3] so that the attacker can fool the authentication system by declaring its corresponding identity. Since the sensor device is inevitably at a direct contact of the user being captured, it is considered one of the weakest point in the entire biometrics system. Because of this, there is a growing interest in automatically analyzing the acquired fingerprint images in order to catch potential malignant users [4]. This kind of attacks are known as presentation attacks [5], and liveness detection techniques are designed to spot them by formulating a binary classification problem with the aim of establishing whether a given biometrics comes from the subject present at the time of capture [6].

The liveness of a fingerprint can be established by designing a software system that analyzes the same images used by the recognition algorithm, or by equipping the scanner with additional hardware. These last prevention measures are called hardware-based systems [7] and are generally more accurate since they take advantage of additional cues. Anyway, the software of a fingerprint scanner can be updated with no additional cost, and if a software technique is robust to a variety of attacks and does not annoy the users with too many false positives, it can be an alternative with regard to acquiring new sensing devices.

Recently, different studies [8–10] have shown the effectiveness of deep learning algorithms for the task of fingerprint liveness detection. Deep learning has seriously improved the state of the art in many fields such as speech recognition, natural language processing, and object recognition [11–13]. The ability to generate hierarchical representations and discover complex structures in raw images allows for better representations with respect to traditional methods based on handcrafted features. Software-based systems for fingerprint liveness detection can take advantage of the very broad literature where similar tasks have already been addressed. Among the recent works, we noticed that it has not yet directly modeled a notion of similarity among real and fake fingerprints that can capture the underlying factors that explain their inter- and intra-class variations. We make a step in this direction by proposing a deep metric learning approach based on Triplet networks [14]. Specifically, these networks map the fingerprint images into a representation space, where the learned distance captures the similarity of the examples coming from the same class and push away the real samples from the fake ones. Unlike other metric learning approaches, such as the ones involving Siamese networks [15], the triplet objective function puts in direct comparison the relation among the classes, giving a notion of context that does not require a threshold selection in order make the prediction [14].

We propose a framework that learns a representation from fingerprint patches, starting from a set of real and fake samples given as a training set. Since at test time only a fingerprint image is given, we make our decision on the basis of a matching score, computed against a set of real and fake patches given as a reference. The similarity metric is learned using an improved version [16, 17] of the original triplet objective formulation [14] that adds a pairwise term that more firmly forces the closeness of two examples of the same class. We performed extensive experiments using ten datasets taken from the fingerprint liveness detection competitions LivDet[1] organized by the Department of Electrical and Electronic Engineering of the University of Cagliari, in cooperation with the Department of Electrical and Computer Engineering of the Clarkson University, held in the years 2009 [7], 2011 [18] and 2013 [19]. We compare our approach with respect to the state of the art, getting competitive performance for all the examined datasets. We also perform the cross-dataset and cross-material experiments, in order to evaluate if the obtained fingerprint representation can be reused in different settings or to spot materials that have not been seen during training.

The chapter is structured as follows. In Sect. 12.2 we present some of the current approaches for designing fingerprint liveness detection systems, and the current state of the art. In Sect. 12.3 we explain the details of the proposed framework and in Sect. 12.4 we provide the experimental results. The final Sect. 12.5 is dedicated to the conclusions.

12.2 Background and Previous Work

In this section, we describe various fingerprint liveness detection techniques, particularly with a focus on the ones related to our method, which can be considered as a static software-based technique [6]. Subsequently, we provide details on some of the most recently proposed software-based approaches in order to contextualize our method and highlight our contributions.

[1]http://livdet.org/.

12.2.1 Background

A first categorization of liveness detection systems can be made by distinguishing between hardware and software systems. As already mentioned, hardware-based systems, also called sensor-based [6], use additional information in order to spot the characteristics of living human fingerprints. Useful clues can be found for instance by detecting the pattern of veins underlying the fingerprints surface [20], measuring the pulse [21], by performing odor analysis [22] and by employing near-infrared sensors [23].

Software, also called feature-based systems, are algorithms that can be introduced into a sensor software in order to add the liveness detection functionality. Our approach falls in this category and can also be considered static, to distinguish it from methods that use multiple images taken during the acquisition process. For instance, dynamic methods can exploit the distortion of the fingertip skin since with respect to gummy materials, it differs in terms of flexibility. In the approach proposed by [24], the user is asked to rotate the finger while touching the sensor. The distortion of the different regions at a direct contact of the sensor are characterized in terms of optical flow and the liveness prediction is based on matching the encoded distortion code sequences over time.

In order to design a software-based system based on machine learning algorithms, a database of real and fake examples of fingerprints is needed. The more spoofing techniques are used, the more the algorithm will be able to generalize to new kind of attacks. In a second categorization of liveness detection systems, we consider how the fingertip pattern is taken from the victim. In the cooperative method, the victim voluntarily puts his/her finger on some workable material that is used to create a mold. From this mold, it is possible to generate high-quality reproductions by filling it with materials such as gelatin, silicone, and wooden glue. Figure 12.1 shows some photographs of artificial fingers. Noncooperative methods instead, capture the scenarios where the pattern has been taken from a latent fingerprint. After taking a high-resolution picture, it is reproduced by generating a three-dimensional surface,

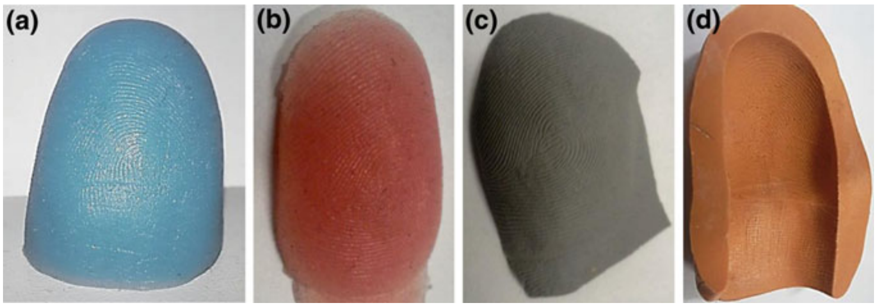


Fig. 12.1 Examples of artificial finger replicas made using different silicon rubber materials: **a** GLS, **b** Ecoflex, **c** Liquid Ecoflex and **d** a Modasil mold

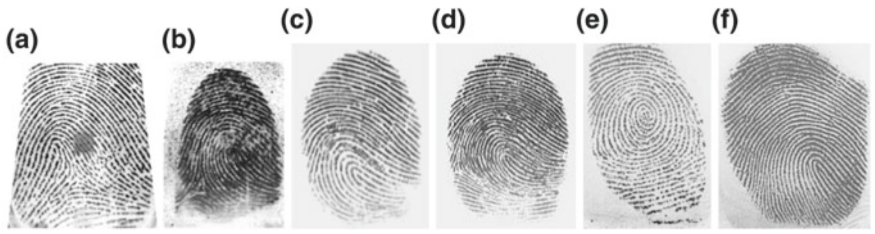


Fig. 12.2 Examples of fake acquisitions from the LivDet 2011 competition (cooperative). From Biometrika **a** Latex, **b** Silicone, from Digital **c** Latex, **d** Gelatine, from Sagem **e** Silicone, **f** Play-doh

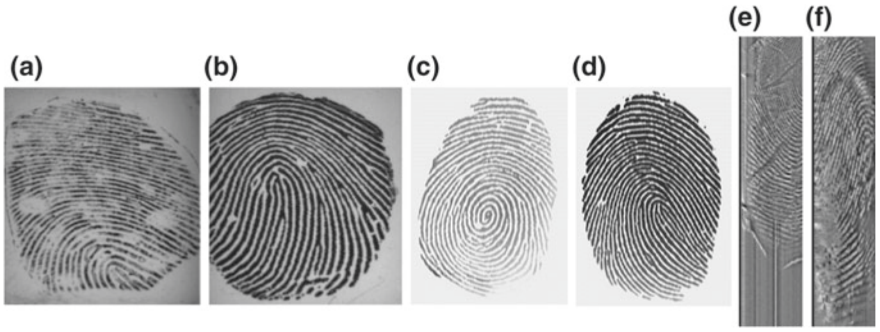


Fig. 12.3 Examples of fake acquisitions from the LivDet 2013 competition. Noncooperative: from Biometrika **a** Gelatine, **b** Wooden Glue, from Italdata **c** Ecoflex, **d** Modasil. Cooperative: from Swipe **e** Latex, **f** Wooden Glue

for instance, by printing it into a circuit board. At this point, a mold can be generated and filled with the above-mentioned materials. The quality of images is inferior as compared to the cooperative methods, and usually, software-based systems have better performance on rejecting these reproductions. Figures 12.2 and 12.3 show several acquisitions, where the fingertip pattern has been captured using the cooperative and noncooperative methods.

12.2.2 Previous Work

In this subsection, we discuss some of the previous work on static software-based fingerprint liveness detection systems. We start by presenting some hand crafted feature-based approaches and conclude with the more recently proposed deep learning techniques.

One of the first approaches to fingerprint liveness detection has been proposed by [25]. It is based on the perspiration pattern of the skin that manifests itself into static and dynamic patterns on the dielectric mosaic structure of the skin. The classification is based on a set of measures extracted from the data and classified using a neural network. In [26], the same phenomenon is captured by a wavelet transform applied to the ridge signal extracted along the ridge mask. After extracting a set of measures from multiple scales, the decision rule is based on classification trees.

Other approaches build some descriptors from different kind of features that are conceived specifically for fingerprint images. In [27] Local binary pattern (LBP) histograms are proposed along with wavelet energy features. The classification is performed by a hybrid classifier composed of a neural network, a support vector machine (SVM) and a k-nearest neighbor classifier. Similar to LBP, the Binary Statistical Image Features (BSIF) [28], encode local textural characteristics into a feature vector and SVM is used for classification. In [29] a Local Contrast Phase Descriptor (LCPD) is extracted by performing image analysis in both the spatial and frequency domains. The same authors propose the use of the Shift-Invariant Descriptor (SID) [30], originally introduced by [31], which provides rotation and scale invariance properties. SID, along with LCPD provides competitive and robust performances on several datasets.

Recently, deep learning algorithms have been applied to fingerprint liveness detection with the aim of automatically finding a hierarchical representation of fingerprints directly from the training data. In [9] the use of convolutional networks has been proposed. In particular, the best results [9] are obtained by fine-tuning the AlexNet and VGG architectures proposed by [11, 32], previously trained on the Imagenet dataset of natural images [33]. From their experimental results, it seems that the factors that most influence the classification accuracy are the depth of the network, the pretraining and the data augmentation they performed in terms of random crops. Since we use a patch-based representation we employ a smaller, but reasonably deep architecture. The use of patches does not require resizing all the images to a fixed dimension, and at the same time, the number of examples is increased so that pretraining can be avoided.

In [8], deep representations are learned from fingerprint, face, and iris images. They are used as traditional features and fed into SVM classifiers to get a liveness score. The authors focus on the choice of the convolutional neural network parameters and architecture.

In [34] deep Siamese networks have been considered along with classical pretrained convolutional networks. This can be considered the most similar work to this chapter since they also learn a similarity metric between a pair of fingerprints. However, their use of metric learning is different since they assume a scenario where the enrollment fingerprint is available for each test image. That is, the decision is made by comparing fingerprints of the same individual. Our approach instead is more versatile and can be applied even if the enrollment fingerprint image is not available.

Different from the above studies, [10, 35] do not give the entire image to the deep learning algorithm but extract patches from the fingerprint acquisition after removing the background. [35] uses classical ConvNets with a binary cross-entropy loss, along with a majority voting scheme to make the final prediction. [10] proposes deep belief networks and use contrastive divergence [36] for pretraining and fine-tunes on the real and fake fingerprint images. The decision is based on a simple threshold applied to the output of the network. Our work is substantially different because it proposes a framework where triplet architectures are used along with a triplet and pairwise objective function.

Summarizing, the contribution of this chapter are (i) a novel deep metric learning based framework, targeted to fingerprint liveness detection, able to work in real time with state-of-the-art performance, (ii) a patch-based and fine-grained representation of the fingerprint images that makes it possible to train a reasonably deep architecture from scratch, even with few hundreds of examples, and that shows superior performance even in settings different from the ones used for training.

12.3 A Deep Triplet Embedding for Fingerprint Liveness Detection

In this section, we describe the proposed method for fingerprint liveness detection based on triplet loss embedding. We start by describing the overall framework; subsequently, we introduce the network architecture and the training algorithm. Finally, we describe the matching procedure that leads to the final decision on the liveness of a given fingerprint image.

12.3.1 Framework

As depicted in Fig. 12.4, the proposed framework requires a collection of real and fake fingerprint images taken from a sensor and used as a training set. From each image, we randomly extract one fixed sized patch and arrange them in a certain number of triplets $\{x_i, x_j^+, x_k^-\}$, where x_i (anchor) and x_j^+ are two examples of the same class, and x_k^- comes from the other class. We alternatively set the anchor to be a real or a fake fingerprint patch.

The architecture is composed of three convolutional networks with shared weights so that three patches can be processed at the same time and mapped into a common feature space. We denote by $\mathbf{r}(\cdot)$ the representation of a given patch obtained from the output of one of the three networks. The deep features extracted from the live and fake fingerprints are compared in order to extract an intra-class distance $d(\mathbf{r}(x), \mathbf{r}(x^+))$ and an inter-class distance $d(\mathbf{r}(x), \mathbf{r}(x^-))$. The objective is to learn d so that the two examples of the same class are closer than two examples taken from different classes, and the distance between two examples of the same class is as short as possible. After training the networks with a certain number of triplets, we extract a new patch from each training sample and generate a new set of triplets. This procedure is repeated until convergence, see more details in Sect. 12.4.2.

After the training procedure is completed, the learned metric is used as a matching distance in order to establish the liveness of a new fingerprint image. Given a query fingerprint, we can extract p (possibly overlapping) patches and give them as input to

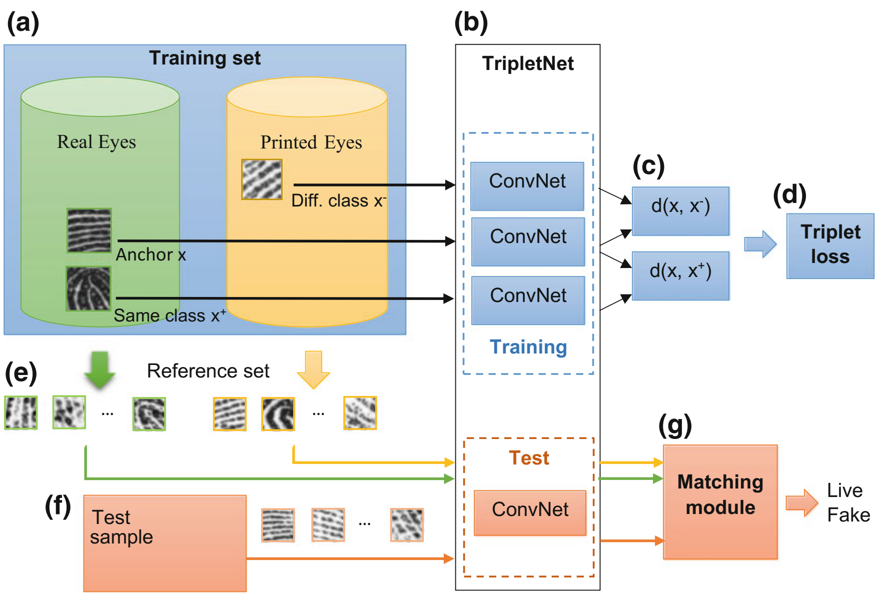


Fig. 12.4 The overall architecture of the proposed fingerprint liveness detection system. From the training set **a** of real and fake fingerprint acquisitions, we train a triplet network **b** using alternatively two patches of one class and one patch of the other one. The output of each input patch is used to compute the inter- and intra-class distances **c** in order to compute the objective function **d** that is used to train the parameters of the networks. After training, a set of real and a set of fake reference patches **e** are extracted from the training set (one for each fingerprint) and the corresponding representation is computed forwarding them through the trained networks. At test time, a set of patches is extracted from the fingerprint image **f** in order to map it to the same representation space as the reference gallery and are matched **g** in order to get a prediction on its liveness

the networks in order to get a representation $Q = \{\mathbf{r}(Q_1), \mathbf{r}(Q_2), \ldots, \mathbf{r}(Q_p)\}$. Since we are not directly mapping each patch to a binary liveness label, but generating a more fine-grained representation, the prediction can be made by a decision rule based on the learned metric d computed with respect to a set R_L and R_F of real and fake reference fingerprints:

$$R_L = \{\mathbf{r}(x_{L_1}), \mathbf{r}(x_{L_2}), \ldots, \mathbf{r}(x_{L_n})\} \tag{12.1a}$$

$$R_F = \{\mathbf{r}(x_{F_1}), \mathbf{r}(x_{F_2}), \ldots, \mathbf{r}(x_{F_n})\} \tag{12.1b}$$

where the patches x_{L_i} and x_{F_i} can be taken from the training set or from a specially made design set.

Table 12.1 Architecture of the proposed embedding network

Layer name	Layer description	Output
input	32×32 gray level image	
conv1	5×5 conv. filters, stride 1, $1 \rightarrow 64$ feat. maps	$64 \times 28 \times 28$
batchnorm1	Batch normalization	$64 \times 28 \times 28$
relu1	Rectifier linear unit	$64 \times 28 \times 28$
conv2	3×3 conv. filters, stride 2, padding 1, $64 \rightarrow 64$ feat. maps	$64 \times 14 \times 14$
conv3	3×3 conv. filters, stride 1, $64 \rightarrow 128$ feat. maps	$64 \times 12 \times 12$
batchnorm2	Batch normalization	$64 \times 12 \times 12$
relu2	Rectifier linear unit	$64 \times 12 \times 12$
conv4	3×3 conv. filters, stride 2, padding 1, $128 \rightarrow 128$ feat. maps	$128 \times 6 \times 6$
conv5	3×3 conv. filters, stride 1, $128 \rightarrow 256$ feat. maps	$256 \times 4 \times 4$
batchnorm3	Batch normalization	$256 \times 4 \times 4$
relu3	Rectifier linear unit	$256 \times 4 \times 4$
conv6	3×3 conv. filters, stride 2, padding 1, $256 \rightarrow 256$ feat. maps	$256 \times 2 \times 2$
fc1	Fully connected layer $4 \times 256 \rightarrow 256$	256
dropout	Dropout $p = 0.4$	256
relu5	Rectifier linear unit	256
fc2	Fully connected layer $256 \rightarrow 256$	256
output	Softmax	256

12.3.2 Network Architecture

We employ a network architecture inspired by [37] where max pooling units, widely used for downsampling purposes, are replaced by simple convolution layers with increased stride. Table 12.1 contains the list of the operations performed by each layer of the embedding networks.

The architecture is composed of a first convolutional layer that takes the 32×32 grayscale fingerprint patches and outputs 64 feature maps by using filters of size 5×5. Then, batch normalization [38] is applied in order to get a faster training convergence and rectified linear units (ReLU) are used as nonlinearities. Another convolutional layer with a stride equal to 2, padding of 1 and filters of size 3×3 performs a downsampling operation by a factor of two in both directions.

The same structure is replicated two times, reducing the filter size to 3×3 and increasing the number of feature maps from 64 to 128 and from 128 to 256. At this point, the feature maps have the size of $128 \times 2 \times 2$ and are further processed by two fully connected layers with 256 outputs followed by a softmax layer. This nonlinearity helps in getting a better convergence of the training algorithm and ensures that the

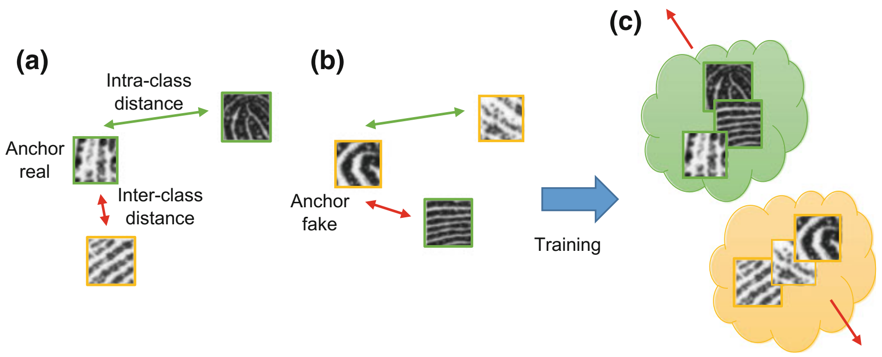


Fig. 12.5 The training procedure uses examples as triplets formed by **a** two real fingerprints (in *green*) and one impostor (in *yellow*) and **b** two impostors and one genuine. The training procedure using the triplet loss will result in an attraction for the fingerprints of the same class (either real or fake) so that their distance will be as close as possible. At the same time, real and fake fingerprints will be pushed away from each other (**c**)

distance among to outputs does not exceed one. Dropout [39] with probability 0.4 is applied to the first fully connected layer for regularization purposes.

The complete network is composed of three instances of this architecture: from three batches of fingerprint images we get the L2 distances between the matching and mismatching images. At test, we take the output of one of the three networks to obtain the representation for a given patch. If there are memory limitations, an alternative consists of using just one network, collapse the three batches into a single one, and computing the distances among the examples corresponding to the training triplets.

12.3.3 Training

As schematized in Fig. 12.5, the triplet architecture along with the triplet loss function aims to learn a metric that makes two patches of the same class closer with respect to two coming from different classes. The objective is to capture the cues that make two fingerprints both real or fake. The real ones come from different people and fingers, and their comparison is performed in order to find some characteristics that make them genuine. At the same time, fake fingerprints come from different people and can be built using several materials. The objective is to detect anomalies that characterize fingerprints coming from a fake replica, without regard to the material they are made of.

Given a set of triplets $\{x_i, x_j^+, x_k^-\}$, where x_i is the anchor and x_j^+ and x_k^- are two examples of the same and the other class, respectively, the objective of the original triplet loss [14] is to give a penalty if the following condition is violated:

$$d(\mathbf{r}(x_i), \mathbf{r}(x_j^+)) - d(\mathbf{r}(x_i), \mathbf{r}(x_k^-)) + 1 \leq 0 \tag{12.2}$$

At the same time, we would like to have the examples of the same class as close as possible so that, when matching a new fingerprint against the reference patches of the same class, the distance $d(\mathbf{r}(x_i), \mathbf{r}(x_j^+))$ is as low as possible. If we denote by $y(x_i)$ the class of a generic patch x_i, we can obtain the desired behavior by formulating the following loss function:

$$L = \sum_{i,j,k} \left\{ c(x_i, x_j^+, x_k^-) + \boldsymbol{\beta} c(x_i, x_j^+) \right\} + \lambda \|\boldsymbol{\theta}\|_2 \tag{12.3}$$

where θ is a one-dimensional vector containing all the trainable parameters of the network, $y(x_i) = y(x_j)$, $y(x_k^-) \neq y(x_i)$ and

$$c(x_i, x_j^+, x_k^-) = \max \left\{ 0, d(\mathbf{r}(x_i), \mathbf{r}(x_j^+)) - d(\mathbf{r}(x_i), \mathbf{r}(x_k^-)) + 1 \right\} \tag{12.4a}$$

$$c(x_i, x_j^+) = d(\mathbf{r}(x_i), \mathbf{r}(x_j^+)) \tag{12.4b}$$

During training, we compute the subgradients and use backpropagation through the network in order to get the desired representation. Contextualizing to what depicted in Fig. 12.5, $c(x_i, x_j^+, x_k^-)$ is the inter-class and $c(x_i, x_j^+)$ the intra-class distance term. $\lambda\|\theta\|_2$ is an additional weight decay term added to the loss function for regularization purposes.

After a certain number of iterations k, we periodically generate a new set of triplets by extracting a different patch from each training fingerprint. It is essential to not update the triplets after too many iterations because it can result in overfitting. At the same time, generating new triplets too often or mining hard examples can cause convergence issues.

12.3.4 Matching

In principle, any distance among bag of features can be used in order to match the query fingerprint $Q = \{\mathbf{r}(Q_1), \mathbf{r}(Q_2), \ldots, \mathbf{r}(Q_p)\}$ against the reference sets R_L and R_F. Since the training objective drastically pushes the distances to be very close to zero or to one, a decision on the liveness can be made by setting a simple threshold $\tau = 0.5$. An alternative could consists of measuring the Hausdorff distance between bags, but it would be too much sensitive to outliers since it involves the computation of the minimum distance between a test patch and each patch of each reference set. Even if using the k-th Hausdorff distance [40], that considers the k-th value instead of the minimum, we obtained better performance by following a simple majority voting strategy. It is also faster since it does not involve sorting out the distances.

Given a fingerprint Q, for each patch Q_j we count how many distances for each reference set are below the given threshold

$$D(R_L, Q_j) = |\{i \in \{1, \ldots, n\} : d(R_{L_i}, Q_j) < \tau\}| \tag{12.5a}$$

$$D(R_F, Q_j) = |\{i \in \{1, \ldots, n\} : d(R_{F_i}, Q_j) < \tau\}| \tag{12.5b}$$

then we make the decision evaluating how many patches belong to the real or the fake class:

$$y(Q) = \begin{cases} \text{real} & \text{if } \sum_{j=1}^{p} D(R_L, Q_j) \geq \sum_{j=1}^{p} D(R_F, Q_j) \\ \text{fake} & \text{otherwise} \end{cases} \tag{12.6}$$

The above method can also be applied in scenarios where multiple fingerprints are acquired from the same individual, as usually happens on passport checks at airports. For instance, the patches coming from different fingers can be accumulated in order to apply the same majority rule of Eq. 12.6 or the decision can be made on the most suspicious fingerprint.

12.4 Experiments

We evaluated the proposed approach with ten of the most popular benchmark for fingerprint liveness detection, coming from the LivDet competitions held in 2009 [7], 2011 [18] and 2013 [19]. We compare our method with the state of the art, specifically the VGG pretrained network of [9], the Local Contrast Phase Descriptor LCPD [29], the dense Scale Invariant Descriptor SID [30] and the Binarized Statistical Image Features [28]. For the main experiments, we strictly follow the competition rules using the training/test splits provided by the organizers while for the cross-dataset and cross-material scenarios, we follow the setup of [9].

The network architecture along with the overall framework have been implemented using the Torch7 computing framework [41] on an NVIDIA® DIGITS™ DevBox with four TITAN X GPUs with seven TFlops of single precision, 336.5 GB/s of memory bandwidth, and 12 GB of memory per board. MATLAB® has been used for image segmentation.

12.4.1 Datasets

The LivDet 2009 datasets [7] were released with the first international fingerprint liveness detection competition, with the aim of becoming a reference and allowing researchers to compare the performance of their algorithms or systems. The fingerprints were acquired using the cooperative approach (see Sect. 12.2.1) and the replicas are created using the materials: gelatin, silicone, and play-doh. The organizers released three datasets, acquired using three different sensors: Biometrika (FX2000), Identix (DFR2100), and Crossmatch (Verifier 300 LC).

Table 12.2 Details of the LivDet 2009 and 2013 competitions. The last row indicates the spoof materials: S = Silicone, G = Gelatine, P = Play-Doh, E = Ecoflex, L = Latex, M = Modasil, B = Body Double, W = Wooden glue

Competition	LivDet2009			LivDet2013		
Dataset	Biometrika	CrossMatch	Identix	Biometrika	Italdata	Swipe
Size	312 × 372	480 × 640	720 × 720	312 × 372	480 × 640	1500 × 208
DPI	569	500	686	569	500	96
Subjects	50	254	160	50	50	100
Live samples	2000	2000	1500	2000	2000	2500
Spoof samples	2000	2000	1500	2000	2000	2000
Materials	S	GPS	GPS	EGLMW	EGLMW	BLPW

The LivDet 2011 competition [18] released four datasets, acquired using the scanners Biometrika (FX2000), Digital Persona (4000B), ItalData (ETT10) and Sagem (MSO300). The materials used for fake fingerprints are gelatin, latex, Ecoflex (platinum-catalyzed silicone), silicone and wooden glue. The spoof fingerprints have been obtained as in LivDet 2009 with the cooperative method.

The LivDet 2013 competition [19] consists of four datasets acquired using the scanners Biometrika (FX2000), ItalData (ETT10), Crossmatch (L SCAN GUARDIAN) and Swipe. Differently from LivDet 2011, two datasets, Biometrika and Italdata, have been acquired using the non-cooperative method. That is, latent fingerprints have been acquired from a surface, and then printed on a circuit board (PCB) in order to generate a three-dimensional structure of the fingerprint that can be used to build a mold. To replicate the fingerprints they used Body Double, latex, PlayDoh and wood glue for the Crossmatch and Swipe datasets and gelatin, latex, Ecoflex, Modasil and wood glue for Biometrika and Italdata.

The size of the images, the scanner resolution, the number of acquired subject and of live and fake samples are detailed in Tables 12.2 and 12.3. The partition of training and test examples is provided by the organizers of the competition.

12.4.2 Experimental Setup

For all the experiments we evaluate performance in terms of average classification error. This is the measure used to evaluate the entries in the LivDet competitions and is the average of the Spoof False Positive Rate (SFPR) and the Spoof False Negative Rate (SFNR). For all the experiments on the LivDet test sets we follow the standard protocol and since a validation set is not provided, we reserved a fixed amount of 120 fingerprints. For the cross-dataset experiments, we used for validation purposes the Biometrika 2009 and Crossmatch 2013 datasets.

Table 12.3 Details of the LivDet 2011 competition. The last row indicates the spoof materials: Sg = Silgum, the others are the same as in Table 12.2

Competition	LivDet2011			
Dataset	Biometrika	Digital persona	Italdata	Sagem
Size	312 × 372	355 × 391	640 × 480	352 × 384
DPI	500	500	500	500
Subjects	200	82	92	200
Live samples	2000	2000	2000	2000
Spoof samples	2000	2000	2000	2000
Materials	EGLSgW	GLPSW	EGLSgW	GLPSW

The triplets set for training is generated by taking one patch from each fingerprint and arranging them alternatively in two examples of one class and one of the other class. The set is updated every $k = 100{,}000$ triplets that are fed to the networks in batches of 100. In the remainder of the chapter, we refer to each update as the start of a new iteration. We use stochastic gradient descent to minimize the triplet loss function, setting a learning rate of 0.5 and a momentum of 0.9. The learning rate η_0 is annealed by following the form:

$$\eta = \frac{\eta_0}{1 + 10^{-4} \cdot b} \tag{12.7}$$

where b is the progressive number of batches that are being processed. That is, after ten iterations the learning rate is reduced by half. The weight decay term of Eq. 12.3 is set to $\lambda = 10^{-4}$ and $\beta = 0.002$ as in [17].

After each iteration, we check the validation error. Instead of using the same accuracy measured at test (the average classification error), we construct 100,000 triplets using the validation set patches, but taking as anchor the reference patches taken from the training set and used to match the test samples. The error consists of the number of violating triplets and reflects how much the reference patches failed to classify patches never seen before. Instead of fixing the number of iterations, we employ early stopping based on the concept of patience [42]. Each time the validation error decrease, we save a snapshot of the network parameters, and if in 20 consecutive iterations the validation error is not decreasing anymore, we stop the training and evaluate the accuracy on the test set using the last saved snapshot.

12.4.3 Preprocessing

Since the images coming from the scanners contain a wide background area surrounding the fingerprint, we segmented the images in order to avoid extracting background patches. The performance is highly affected by the quality of the background subtraction, therefore, we employed an algorithm [43], that divides the fingerprint image into 16 × 16 blocks, and classifies a block as foreground only if its standard deviation is more than a given threshold. The rationale is that a higher standard deviation corresponds to the ridge regions of a fingerprint. In order to exclude background noise that can interfere with the segmentation, we compute the connected components of the foreground mask and take the fingerprint region as the one with the largest area. In order to get a smooth segmentation, we generate the convex hull image from the binary mask using morphological operations.

We also tried to employ data augmentation techniques in terms of random rotations, flipping and general affine transformation. Anyway, they significantly slowed down the training procedure and we did not get any performance improvement on either the main and cross-dataset experiments.

12.4.4 Experimental Results

In this section, we present the performance of the proposed fingerprint liveness detection system in different scenarios. In Table 12.4 we list the performance in terms of average classification error on the LivDet competition test sets. With respect to the currently best-performing methods [9, 29, 30] we obtained competitive performance for all the datasets, especially on Italdata 2011, and Swipe 2013. This means that the approach works properly also on the images coming from swipe scanners, where the fingerprints are acquired by swiping the finger across the sensor surface (see Fig. 12.3e, f). Overall, our approach has an average error of 1.75% in comparison

Table 12.4 Average classification error for the LivDet Test Datasets. In column 2 our TripletNet based approach, in column 2 the VGG deep network pretrained on the Imagenet dataset and fine-tuned by [9], in column 3 the Local Contrast Phase Descriptor [29] based approach, in column 4 the dense Scale Invariant Descriptor [30] based approach and in column 5 the Binarized Statistical Image Features [28] based approach

Dataset	TripletNet	VGG [9]	LCPD [29]	SID [30]	BSIF [28]
Biometrika 2009	**0.71**	4.1	1	3.8	9
CrossMatch 2009	1.57	**0.6**	3.4	3.3	5.8
Identix 2009	**0.044**	0.2	1.3	0.7	0.7
Biometrika 2011	5.15	5.2	**4.9**	5.8	6.8
Digital 2011	**1.85**	3.2	4.7	2.0	4.1
Italdata 2011	**5.1**	8	12.3	11.2	13.9
Sagem 2011	**1.23**	1.7	3.2	4.2	5.6
Biometrika 2013	**0.65**	1.8	1.2	2.5	1.1
Italdata 2013	0.5	**0.4**	1.3	2.7	3
Swipe 2013	**0.66**	3.7	4.7	9.3	5.2
Average	**1.75%**	2.89%	3.8%	4.5%	5.5%

to the 2.89% of [9] which results in a performance improvement of 65%. We point out that we did not use the dataset CrossMatch 2013 for evaluation purposes because the organizers of the competition found anomalies in the data and discouraged its use for comparative evaluations [4]. In Fig. 12.6 we depict a 2D representation of the test set of Biometrika 2013, specifically one patch for every fingerprint image, computed from an application of t-SNE [44] to the generated embedding. This dimensionality reduction technique is particularly insightful since it maps the high-dimensional representation in a space where the vicinity of points is preserved. We can see that the real and fake fingerprints are well separated and only a few samples are in the wrong place, for the major part Wooden Glue and Modasil. Ecoflex and gelatin replicas seem more easy to reject. Examining the patch images, we can see that going top to bottom, the quality of the fingerprint pattern degrades. This may be due to the perspiration of the fingertips that makes the ridges not as uniform as the fake replicas.

12.4.4.1 Cross-Dataset Evaluation

As in [9], we present some cross-dataset evaluation and directly compare our performance with respect to their deep learning and Local Binary Pattern approach. The results are shown in Table 12.5 and reflect a significant drop in performance with respect to the previous experiments. With respect to [9] the average classification error is slightly better, anyway it is too high to possibly consider doing liveness detection in the wild. Similar results have been obtained by [34]. We point out that different sensors, settings and climatic conditions can extremely alter the fingerprint images, and if the training set is not representative of the particular conditions, any machine learning approach, not just deep learning algorithms, would not be effective at generalization.

12.4.4.2 Cross-Material Evaluation

We also evaluated the robustness of our system to new spoofing materials. We followed the protocol of [9] by training the networks using a subset of materials and testing on the remaining ones. The results are shown in Table 12.6. With respect to the cross-dataset experiments, the method appears to be more robust to new materials rather than a change of the sensor. Also in this scenario, if we exclude the Biometrika 2011 dataset, our approach has a significative improvement with respect to [9].

12.4.4.3 Computational Efficiency

One of the main benefits of our approach is the computational time since the architecture we employed is smaller in comparison to other deep learning approaches such as [9, 34]. Moreover, the patch representation allows for scaling the matching

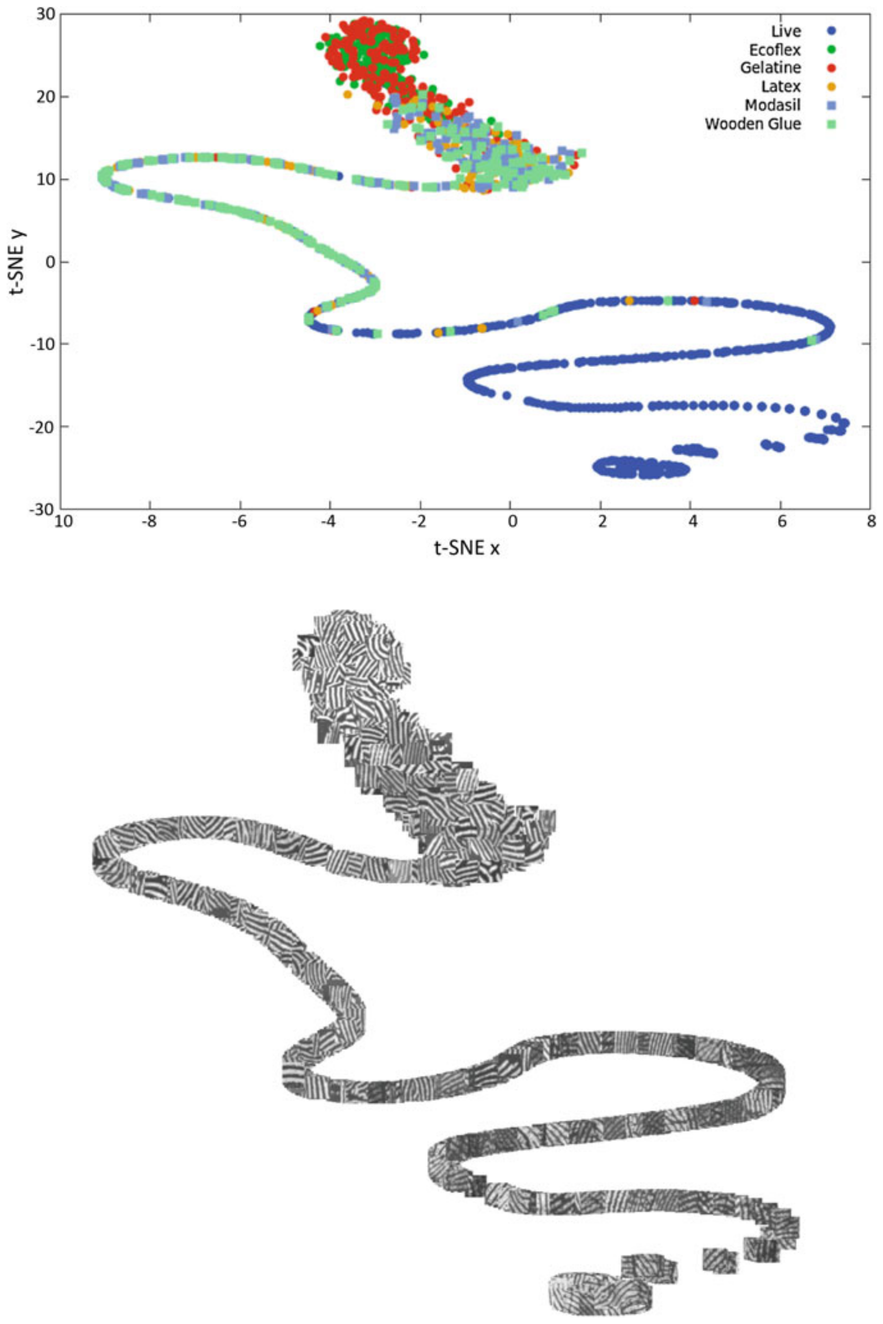


Fig. 12.6 T-SNE visualization of the embedding generated from the live and fake fingerprints composing the test set of Biometrika 2013 (one patch for each acquisition). The high dimensional representation is mapped into a two-dimensional scatter plot where the vicinity of points is preserved

Table 12.5 Average classification error for the cross-dataset scenarios. The first column is the dataset used for training and the second the one used for the test. The third column is our TripletNet approach, the fourth and the fifth are the deep learning and the Local Binary Patterns (LBP) based approaches proposed by [9]

Training set	Test set	TripletNet	VGG	LBP
Biometrika 2011	Biometrika 2013	14	15.5	16.5
Biometrika 2013	Biometrika 2011	34.05	46.8	47.9
Italdata 2011	Italdata 2013	8.3	14.6	10.6
Italdata 2013	Italdata 2011	44.65	46.0	50.0
Biometrika 2011	Italdata 2011	29.35	37.2	47.1
Italdata 2011	Biometrika 2011	27.65	31.0	49.4
Biometrika 2013	Italdata 2013	1.55	8.8	43.7
Italdata 2013	Biometrika 2013	3.8	2.3	48.4

Table 12.6 Average Classification Error for the cross-material scenario. In column 2 are the materials used for training and in column 3 the ones used for the test. The abbreviations are the same as in Tables 12.2 and 12.3

Dataset	Train materials	Test materials	TripletNet	VGG	LBP
Biometrika 2011	EGL	SgW	13.1	10.1	17.7
Biometrika 2013	MW	EGL	2.1	4.9	8.5
Italdata 2011	EGL	SgW	7	22.1	30.9
Italdata 2013	MW	EGL	1.25	6.3	10.7

procedure on different computational units, so that it can be used also in heavily populated environments. In our experiments, we extract 100 patches from each test fingerprint and the time to get their corresponding representation is about 0.6ms using a single GPU and 0.3 s using a Core i7-5930K 6 Core 3.5 GHz desktop processor (single thread). Considering the most common dataset configuration of 880 real and 880 fake reference patches, the matching procedure takes 5.2 ms on a single GPU and 14 ms on the CPU. Finally, the training time varies depending on the particular dataset, and on the average, the procedure converges in 135 iterations. A single iteration takes 84 and 20 s are needed to check the validation error.

12.5 Conclusions

In this chapter, we introduced a novel framework for fingerprint liveness detection which embeds the recent advancements in deep metric learning. We validated the effectiveness of our approach in a scenario where the fingerprints are acquired using the same sensing devices that are used for training. We also presented quantified results on the generalization capability of the proposed approach for new acquisition devices, and unseen spoofing materials. The approach is able to work in real time and surpasses the state-of-the-art on several benchmark datasets.

In conclusion, we point out that the employment of software-based liveness detection systems should never give a sense of false security to their users. As in other areas such as cyber-security, the attackers become more resourceful every day and new ways to fool a biometric system will be discovered. Therefore, such systems should be constantly updated and monitored, especially in critical applications such as airport controls. It would be desirable to have large datasets that contain fingerprint images of people with different age, sex, ethnicity, and skin conditions and that are acquired under different time periods, environments and using a variety of sensors with a multitude of spoofing materials.

Acknowledgements This work was supported in part by grant N00014-12-1-1026.

References

1. A.K. Jain, A. Ross, S. Pankanti, Biometrics: a tool for information security. IEEE Trans. Inf. Forensics Secur. **1**(2), 125–143 (2006)
2. M. Martinez-Diaz, J. Fierrez, J. Galbally, J. Ortega-Garcia, An evaluation of indirect attacks and countermeasures in fingerprint verification systems. Pattern Recogn. Lett. **32**(12), 1643–1651 (2011)
3. T. Matsumoto, H. Matsumoto, K. Yamada, S. Hoshino, Impact of artificial "gummy" fingers on fingerprint systems. Proc. Conf. Opt. Secur. Counterfeit Deterrence Tech. **4677**, 275–289 (2002)
4. L. Ghiani, D.A. Yambay, V. Mura, G.L. Marcialis, F. Roli, S.A. Schuckers, Review of the fingerprint liveness detection (livdet) competition series: 2009 to 2015. *Image and Vision Computing* (2016) (in press)
5. C. Sousedik, C. Busch, Presentation attack detection methods for fingerprint recognition systems: a survey. IET Biom. **3**(4), 219–233 (2014)
6. J. Galbally, S. Marcel, J. Fierrez, Biometric antispoofing methods: a survey in face recognition. IEEE Access **2**, 1530–1552 (2014)
7. G.L. Marcialis, A. Lewicke, B. Tan, P. Coli, D. Grimberg, A. Congiu, A. Tidu, F. Roli, S. Schuckers, First international fingerprint liveness detection competition – livdet 2009, in *Proceedings of the International Conference on Image Analysis and Processing (ICIAP, 2009)* (Springer, Berlin, 2009), pp. 12–23
8. D. Menotti, G. Chiachia, A. Pinto, W.R. Schwartz, H. Pedrini, A.X. Falcao, A. Rocha, Deep representations for iris, face, and fingerprint spoofing detection. IEEE Trans. Inf. Forensics Secur. **10**(4), 864–879 (2015)
9. R.F. Nogueira, R. de Alencar Lotufo, R.C. Machado, Fingerprint liveness detection using convolutional neural networks. IEEE Trans. Inf. Forensics Secur. **11**(6), 1206–1213 (2016)
10. S. Kim, B. Park, B.S. Song, S. Yang, Deep belief network based statistical feature learning for fingerprint liveness detection. Pattern Recogn. Lett. **77**, 58–65 (2016)
11. A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in *Conference on Advances in Neural Information Processing Systems (NIPS, 2012)* (2012), pp. 1097–1105
12. Y. LeCun, Y. Bengio, G. Hinton, Deep learning. Nature **521**(7553), 436–444 (2015)
13. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning* (MIT Press, Cambridge, 2016), http://www.deeplearningbook.org

14. E. Hoffer, N. Ailon, Deep metric learning using triplet network, in *Proceedings of the International Workshop on Similarity-Based Pattern Recognition (SIMBAD, 2015)* (Springer, Berlin, 2015), pp. 84–92
15. J. Bromley, J.W. Bentz, L. Bottou, I. Guyon, Y. LeCun, C. Moore, E. Säckinger, R. Shah, Signature verification using a “siamese” time delay neural network. Int. J. Pattern Recognit. Artif. Intell. **7**(04), 669–688 (1993)
16. P. Wohlhart, V. Lepetit, Learning descriptors for object recognition and 3D pose estimation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2015)* (2015), pp. 3109–3118
17. D. Cheng, Y. Gong, S. Zhou, J. Wang, N. Zheng, Person re-identification by multi-channel parts-based cnn with improved triplet loss function, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR, 2016)* (2016)
18. D. Yambay, L. Ghiani, P. Denti, G.L. Marcialis, F. Roli, S. Schuckers, Livdet 2011 - fingerprint liveness detection competition 2011, in *Proceedings of the IAPR International Conference on Biometrics (ICB, 2011)* (2012), pp. 208–215
19. L. Ghiani, D. Yambay, V. Mura, S. Tocco, G.L. Marcialis, F. Roli, S. Schuckcrs, Livdet 2013 fingerprint liveness detection competition 2013, in *Proceedings of the International Conference on Biometrics (ICB, 2013)* (2013), pp. 1–6
20. A. Kumar, Y. Zhou, Human identification using finger images. IEEE Trans. Image Process. **21**(4), 2228–2244 (2012)
21. K. Seifried, Biometrics-what you need to know. Secur. Portal **10** (2001)
22. D. Baldisserra, A. Franco, D. Maio, D. Maltoni, Fake fingerprint detection by odor analysis, in *Proceedings of the International Conference on Biometrics (ICB, 2006)* (Springer, Berlin, 2006), pp. 265–272
23. P.V. Reddy, A. Kumar, S.M.K. Rahman, T.S. Mundra, A new antispoofing approach for biometric devices. IEEE Trans. Biomed. Circuits Syst. **2**(4), 328–337 (2008)
24. A. Antonelli, R. Cappelli, D. Maio, D. Maltoni, Fake finger detection by skin distortion analysis. IEEE Trans. Inf. Forensics Secur. **1**(3), 360–373 (2006)
25. R. Derakhshani, S.A.C. Schuckers, L.A. Hornak, L. O’Gorman, Determination of vitality from a non-invasive biomedical measurement for use in fingerprint scanners. Pattern Recogn. **36**(2), 383–396 (2003)
26. B. Tan, S. Schuckers, Liveness detection for fingerprint scanners based on the statistics of wavelet signal processing, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPRW, 2006)* (IEEE, 2006), p. 26
27. S.B. Nikam, S. Agarwal, Texture and wavelet-based spoof fingerprint detection for fingerprint biometric systems, in *Proceedings of the International Conference on Emerging Trends in Engineering and Technology (ICETET, 2008)* (IEEE, 2008), pp. 675–680
28. L. Ghiani, A. Hadid, G.L. Marcialis, F. Roli, Fingerprint liveness detection using binarized statistical image features, in *Proceedings of the International Conference on Biometrics Theory, Applications and Systems (BTAS, 2013)* (IEEE, 2013), pp. 1–6
29. D. Gragnaniello, G. Poggi, C. Sansone, L. Verdoliva, Local contrast phase descriptor for fingerprint liveness detection. Pattern Recogn. **48**(4), 1050–1058 (2015)
30. D. Gragnaniello, G. Poggi, C. Sansone, L. Verdoliva, An investigation of local descriptors for biometric spoofing detection. IEEE Trans. Inf. Forensics Secur. **10**(4), 849–863 (2015)
31. I. Kokkinos, M. Bronstein, A. Yuille, Dense Scale Invariant Descriptors for Images and Surfaces. Research report RR-7914, INRIA (2012)
32. K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition (2014), arXiv.org/abs/1409.1556
33. R. Olga, D. Jia, S. Hao, K. Jonathan, S. Sanjeev, M. Sean, H. Zhiheng, K. Andrej, K. Aditya, B. Michael, C.B. Alexander, F. Li, Imagenet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
34. E. Marasco, P. Wild, B. Cukic, Robust and interoperable fingerprint spoof detection via convolutional neural networks (hst 2016), in *Proceedings of the IEEE Symposium on Technologies for Homeland Security* (2016), pp. 1–6

35. C. Wang, K. Li, Z. Wu, Q. Zhao, *A DCNN Based Fingerprint Liveness Detection Algorithm with Voting Strategy* (Springer International Publishing, Cham, 2015). pp. 241–249
36. G.E. Hinton, Training products of experts by minimizing contrastive divergence. Neural Comput. **14**(8), 1771–1800 (2002)
37. J.T. Springenberg, A. Dosovitskiy, T. Brox, M. Riedmiller, Striving for simplicity: the all convolutional net (2014), arXiv.org/abs/1412.6806
38. S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in *Proceedings of the International Conference on Machine Learning (ICML, 2015)* (2015), p. 37
39. N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. **15**(1), 1929–1958 (2014)
40. J. Wang, J. Zucker, Solving multiple-instance problem: a lazy learning approach (2000), pp. 1119–1126
41. R. Collobert, K. Kavukcuoglu, C. Farabet, Torch7: a matlab-like environment for machine learning, in *BigLearn, NIPS Workshop* (2011)
42. Y. Bengio, Practical recommendations for gradient-based training of deep architectures, in *Neural Networks: Tricks of the Trade* (Springer, Berlin, 2012), pp. 437–478
43. P.D. Kovesi, MATLAB and Octave functions for computer vision and image processing (2005), http://www.peterkovesi.com/matlabfns
44. L. van der Maaten, G. Hinton, Visualizing data using t-sne. J. Mach. Learn. Res. **9**, 2579–2605 (2008)

Index